

File 256:SoftBase:Reviews,Companies&Prods. 82-2004/Jan  
(c)2004 Info.Sources Inc  
File 2:INSPEC 1969-2004/Feb W1  
(c) 2004 Institution of Electrical Engineers  
File 6:NTIS 1964-2004/Feb W2  
(c) 2004 NTIS, Intl Cpyrght All Rights Res  
File 8:Ei Compendex(R) 1970-2004/Jan W4  
(c) 2004 Elsevier Eng. Info. Inc.  
File 34:SciSearch(R) Cited Ref Sci 1990-2004/Feb W1  
(c) 2004 Inst for Sci Info  
File 35:Dissertation Abs Online 1861-2004/Jan  
(c) 2004 ProQuest Info&Learning  
File 65:Inside Conferences 1993-2004/Feb W2  
(c) 2004 BLDSC all rts. reserv.  
File 94:JICST-EPlus 1985-2004/Feb W1  
(c)2004 Japan Science and Tech Corp(JST)  
File 95:TEME-Technology & Management 1989-2004/Jan W4  
(c) 2004 FIZ TECHNIK  
File 99:Wilson Appl. Sci & Tech Abs 1983-2004/Jan  
(c) 2004 The HW Wilson Co.  
File 111:TGG Natl.Newspaper Index(SM) 1979-2004/Feb 05  
(c) 2004 The Gale Group  
File 144:Pascal 1973-2004/Feb W1  
(c) 2004 INIST/CNRS  
File 202:Info. Sci. & Tech. Abs. 1966-2004/Jan 20  
(c) 2004 EBSCO Publishing  
File 233:Internet & Personal Comp. Abs. 1981-2003/Sep  
(c) 2003 EBSCO Pub.  
File 266:FEDRIP 2004/Dec  
Comp & dist by NTIS, Intl Copyright All Rights Res  
File 434:SciSearch(R) Cited Ref Sci 1974-1989/Dec  
(c) 1998 Inst for Sci Info  
File 483:Newspaper Abs Daily 1986-2004/Feb 06  
(c) 2004 ProQuest Info&Learning  
File 583:Gale Group Globalbase(TM) 1986-2002/Dec 13  
(c) 2002 The Gale Group  
File 603:Newspaper Abstracts 1984-1988  
(c)2001 ProQuest Info&Learning

Set	Items	Description
S1	169	MUTEX? OR MUTUAL()EXCUSION? OR MUT()EX OR MUTUALEXCLUSION?
S2	8164537	THREAD???? ? OR PROCESS OR PROCESSES OR PROCESSING OR TASK- ???? ?
S3	96397	(PLURALITY OR MANY OR MULTI OR SEVERAL OR NUMEROUS OR MULT- IPLE OR MULTITUD? OR MULTIPLICITY OR PLURIF? OR VARIOUS OR VA- RIETY) (1W)S2
S4	77864	(GROUP???? ? OR CLUSTER??? ? OR NUMBER OR NETWORK? ? OR CH- AIN? ? OR SERIES OR THREE OR COLLECTION? OR THIRD OR ASSORT? - OR RANGE? ?) (1W)S2
S5	53704	MULTITHREAD? OR MULTIPROCESS OR MULTIPROCESSES OR MULTIPRO- CESSING OR MULTITASK?
S6	1	GHOSTLOCK? OR GHOST()LOCK??? ?
S7	1064342	OWN OR OWNS OR OWNER? OR OWNED
S8	9844	OWNING
S9	7688	CO()S7:S8 OR S7:S8(2N) (TEMPORAR??? ? OR BRIEF?? ? OR INTER- IM OR MOMENTAR? OR TRANSIT?)
S10	7074	S7:S8(2N) (PARTIAL? OR PORTION? OR PART OR DIVID?? ? OR DIV- ISION? OR SUBDIVID? OR SUBDIVISION? OR PARTITION? OR REDIVID? OR REDIVISION? OR SPLIT???? ?)
S11	32	PARTOWNER? OR COOWN?
S12	42	S1 AND S3:S5

S13	0	S12 AND (S6 OR S9:S11)
S14	15	S12/2001:2004
S15	27	S12 NOT S14
S16	24	RD (unique items)
S17	4525	MUTUAL(1W)EXCLUSION?
S18	2412	(PLURALITY OR MANY OR MULTI OR SEVERAL OR NUMEROUS OR MULTIPLE OR MULTITUD? OR MULTIPLICITY OR PLURIF? OR VARIOUS OR VARIETY) (1W)S7:S8
S19	8497	(GROUP???? ? OR CLUSTER??? ? OR NUMBER OR NETWORK? ? OR CHAIN? ? OR SERIES OR THREE OR COLLECTION? OR THIRD OR ASSORT? - OR RANGE? ?) (1W)S7:S8
S20	674	(S1 OR S17) AND S3:S5
S21	0	S20 AND (S6 OR S9:S11 OR S18:S19)
S22	22	S20 AND S7:S8
S23	5	S22/2001:2004
S24	14	S22 NOT (S23 OR S12)
S25	9	RD (unique items)
S26	0	(S1 OR S17) AND (S9:S11 OR S18:S19)

16/7/1 (Item 1 from file: 2)  
DIALOG(R)File 2:INSPEC  
(c) 2004 Institution of Electrical Engineers. All rts. reserv.

6769781 INSPEC Abstract Number: C2001-01-6110P-009

**Title: A multi - threaded server for shared hash table access**

Author(s): Vckovski, A.; Brazile, J.

Conference Title: Proceedings of the 7th USENIX Tcl/Tk Conference  
(Tcl/2K) p.175-82

Publisher: USENIX Assoc, Berkeley, CA, USA

Publication Date: 2000 Country of Publication: USA 194 pp.

ISBN: 1 880446 24 3 Material Identity Number: XX-2000-00391

Conference Title: Proceedings of Tcl/2K: 7th USENIX Tcl/Tk Conference

Conference Date: 14-18 Feb. 2000 Conference Location: Austin, TX, USA

Language: English Document Type: Conference Paper (PA)

Treatment: Practical (P)

Abstract: This paper presents a **multi - threaded** socket server allowing access to shared hash tables. It is implemented using Tcl 8.1 **multi - threading** capabilities and runs multiple Tcl interpreters to service client requests. The application is designed as a pre-threaded server which allows a single working thread to handle many requests. The central shared data object is a hash table with structured values which allows access by all threads. Synchronization is based on a reader/writer lock implementation using the synchronization primitives available in Tcl, i.e., **mutexes** and condition variables. The application achieves insert rates that are significantly higher than what current commercial database management systems achieve. The usage of third-level language programming in C and application-specific scripting in Tcl allows a design based on a lightweight robust kernel on the one hand and easily modifiable application-domain code on the other. The experiences with thread safety and other threading features in Tcl 8.1 have been largely positive in this real-world application. (5 Refs)

Subfile: C

Copyright 2000, IEE

16/7/2 (Item 2 from file: 2)  
DIALOG(R)File 2:INSPEC  
(c) 2004 Institution of Electrical Engineers. All rts. reserv.

6694227 INSPEC Abstract Number: C2000-10-6110J-035

**Title: SIPO: Simple Inter-Process Objects**

Author(s): Donovan, S.

Journal: EXE vol.15, no.3 p.14-18

Publisher: Centaur Communications,

Publication Date: Aug. 2000 Country of Publication: UK

CODEN: EXEEE5 ISSN: 0268-6872

SICI: 0268-6872(200008)15:3L.14:SSIP;1-F

Material Identity Number: L815-2000-008

Language: English Document Type: Journal Paper (JP)

Treatment: Practical (P)

Abstract: In this practical example of real-world C++ usage, the author presents a straightforward and compact model for Win32-based inter-process communication. He shows how a simple form of inter-process communication (SIPO) can be set up for Win32 that is straightforward, compact and fast.

It is specifically tailored to C++ and supports the standard library strings and containers. Along the way, he covers topics like memory-mapped files, threads and synchronisation objects like **mutexes**, which are interesting and useful in their own right. (0 Refs)

Subfile: C

Copyright 2000, IEE

16/7/3 (Item 3 from file: 2)

DIALOG(R)File 2:INSPEC

(c) 2004 Institution of Electrical Engineers. All rts. reserv.

6635730 INSPEC Abstract Number: C2000-08-0220-033

**Title: The design and construction of a user-level kernel for teaching multithreaded programming**

Author(s): Bedy, M.J.; Carr, S.; Xianglong Huang; Ching-Kuang Shene

Author Affiliation: Dept. of Comput. Sci., Michigan Technol. Univ., Houghton, MI, USA

Conference Title: FIE'99 Frontiers in Education. 29th Annual Frontiers in Education Conference. Designing the Future of Science and Engineering Education. Conference Proceedings (IEEE Cat. No.99CH37011 Part vol.2 p.13A3/24-9 vol.2

Publisher: Stripes Publishing L.L.C, Champaign, IL, USA

Publication Date: 1999 Country of Publication: USA 3 vol. 1958 pp.

ISBN: 0 7803 5643 8 Material Identity Number: XX-2000-00612

U.S. Copyright Clearance Center Code: 0 7803 5643 8/99/\$10.00

Conference Title: Proceedings of IEEE Computer Society Conference on Frontiers in Education

Conference Sponsor: IEEE Educ. Soc.; IEEE Comput. Soc.; ASEE Educ. Res. & Methods Div

Conference Date: 10-13 Nov. 1999 Conference Location: San Juan, Puerto Rico

Language: English Document Type: Conference Paper (PA)

Treatment: Practical (P)

**Abstract:** **Multithreading** is a powerful programming paradigm that has become very popular. The authors have developed a set of course materials and software tools for effectively teaching **multithreaded** programming (MTP). One important component of the authors' system is a very simple user-level kernel for instructors to teach MTP without getting into system details, and for the students to add extensions. This paper presents the design and implementation of this kernel as well as its use in the classroom. This minimal user-level kernel employs a first-come-first-served scheduling policy, and permits a user to create and join threads, and use **mutex** locks. With this kernel, students are able to implement semaphores, barriers, reader-writer locks, mail-boxes and condition variables. This approach has two advantages: (1) students can easily learn the basics and internals of a kernel that supports MTP, and (2) conventional debuggers can be used for debugging purposes, because the kernel is a user-level program.

(11 Refs)

Subfile: C

Copyright 2000, IEE

16/7/4 (Item 4 from file: 2)

DIALOG(R)File 2:INSPEC

(c) 2004 Institution of Electrical Engineers. All rts. reserv.

6539129 INSPEC Abstract Number: B2000-05-8110C-009, C2000-05-7410B-011

**Title: Multi - task processing in the adaptive emergency control system [for power systems]**

Author(s): Zhu Kun; Fang Yongjie; Fan Wentao; Xue Yusheng  
Author Affiliation: Nanjing Autom. Res. Inst., China  
Journal: Automation of Electric Power Systems vol.24, no.3 p.42-4,

59

Publisher: Automation of Electric Power Systems Press,  
Publication Date: 10 Feb. 2000 Country of Publication: China  
CODEN: DXZIE9 ISSN: 1000-1026  
SICI: 1000-1026(20000210)24:3L:42:MTPA;1-7  
Material Identity Number: C804-2000-006  
Language: Chinese Document Type: Journal Paper (JP)  
Treatment: Applications (A); Practical (P)

Abstract: The proposed online pre-decision emergency control system contains **many** parallel **tasks** running in several computers, and there are a lot of interchanged messages and data among tasks and among computers. Therefore, it is required to manage and coordinate the processes as well as information exchange for reliability and efficiency. The manager module is based on the following mechanisms: (1) a supervisor thread manages all other threads by using thread starting, suspending, resuming, terminating and priority setting. As for **several threads** with the same priority value, a FIFO algorithm is used; (2) monitor based synchronization and **mutex** processing instead of P, V operation is applied to meet the requirements of emergency control; and (3) watchdog detecting and multi-layer restarting mechanisms are utilized for deadlock processing. The effectiveness of the above mechanisms has been verified with test records.

(2 Refs)

Subfile: B C

Copyright 2000, IEE

16/7/5 (Item 5 from file: 2)

DIALOG(R)File 2:INSPEC

(c) 2004 Institution of Electrical Engineers. All rts. reserv.

6516851 INSPEC Abstract Number: C2000-04-6110J-019

**Title: Making primitive objects thread safe**

Author(s): Lowy, J.

Journal: C/C++ Users Journal vol.18, no.3 p.85-6

Publisher: Miller Freeman,

Publication Date: March 2000 Country of Publication: USA

CODEN: CCUJEX ISSN: 1075-2838

SICI: 1075-2838(200003)18:3L:85:MPOT;1-Q

Material Identity Number: H359-2000-003

Language: English Document Type: Journal Paper (JP)

Treatment: Practical (P)

Abstract: All sorts of things need thread locks. A fairly simple template or two can do the job. **Multithreading** can bring many advantages to an application: the user interface can remain responsive while processing goes on in the background; the application can take advantages of multiple CPUs, serve multiple clients, and prioritize tasks. But these benefits come with a price: you have to worry about synchronization issues, deadlocks, reentrancy, and managing the state of objects that are being accessed on **multiple threads**. The solution presented is a template class called SAFETYPE. SAFETYPE provides synchronization at the atomic level, both for primitive types and for complex classes such as CString. SAFETYPE implements almost all of the C operators. It has only two member variables: an object of type T (the type specified by the template parameter), and a lock (a Win32 **mutex**). (0 Refs)

Subfile: C

Copyright 2000, IEE

16/7/6 (Item 6 from file: 2)  
DIALOG(R)File 2:INSPEC  
(c) 2004 Institution of Electrical Engineers. All rts. reserv.

6109762 INSPEC Abstract Number: C9901-6150J-023

**Title:** Mutexes prevent priority inversions

Author(s): Kalinsky, D.

Author Affiliation: Integrated Syst., Sunnyvale, CA, USA

Journal: Embedded Systems Programming vol.11, no.8 p.76-8, 80-1

Publisher: Miller Freeman,

Publication Date: Aug. 1998 Country of Publication: USA

CODEN: EYPRE4 ISSN: 1040-3272

SICI: 1040-3272(199808)11:8L:76:MPPI;1-0

Material Identity Number: 0692-98009

Language: English Document Type: Journal Paper (JP)

Treatment: Practical (P)

Abstract: Priority inversions can interfere with the normal operation of multitasking application software. The use of mutexes, which provide mutual exclusion for data structure access, can solve the problem. After I describe the phenomena of priority inversion, I present a detailed example of a priority inversion scenario. I make the point that even when using an operating system's semaphore features, a serious problem known as unbounded priority inversion can develop. I introduce an operating system mechanism called a mutex (for mutual exclusion), which has the potential of addressing problems of unbounded priority inversion. Mutexes come in several varieties. Priority inheritance mutexes and priority ceiling mutexes are examined in detail, and their relative advantages and disadvantages weighed. (2 Refs)

Subfile: C

Copyright 1998, IEE

16/7/7 (Item 7 from file: 2)  
DIALOG(R)File 2:INSPEC  
(c) 2004 Institution of Electrical Engineers. All rts. reserv.

5871931 INSPEC Abstract Number: C9805-6150E-001

**Title:** Windows CE Win32 API programming

Author(s): Radtke, B.

Journal: Dr. Dobbs's Journal vol.23, no.4 p.62, 64, 66, 68, 70

Publisher: Miller Freeman,

Publication Date: April 1998 Country of Publication: USA

CODEN: DDJSMD ISSN: 1044-789X

SICI: 1044-789X(199804)23:4L:62:WWP;1-X

Material Identity Number: B719-98003

Language: English Document Type: Journal Paper (JP)

Treatment: Practical (P)

Abstract: Windows CE provides a multitasking pre-emptive environment that manages threads, processes, and resources. It provides synchronization objects such as critical sections, mutexes, and events (semaphores, however, are not supported). It schedules the execution of a thread based on a priority time-sliced algorithm. Threads with the same priority are selected on a round-robin basis. Priorities are fixed and are not changed by the scheduler (in other words, there is no aging algorithm). Priorities can change if the CE scheduler detects a priority inversion problem. The Windows CE kernel supports a maximum of 32 processes. The author examines Windows CE and its development environment, focusing on the differences between version 1.0 and 2.0. (0 Refs)

Subfile: C

Copyright 1998, IEE

16/7/8 (Item 8 from file: 2)  
DIALOG(R)File 2:INSPEC  
(c) 2004 Institution of Electrical Engineers. All rts. reserv.

5576061 INSPEC Abstract Number: C9706-6110J-033

**Title: Thread synchronization**

Author(s): Onion, F.; Harrison, A.  
Author Affiliation: DevelopMentor, ME, USA  
Journal: C++ Report vol.9, no.5 p.57-62  
Publisher: SIGS Publications,  
Publication Date: May 1997 Country of Publication: USA  
CODEN: CRPTE7 ISSN: 1040-6042  
SICI: 1040-6042(199705)9:5L:57:TS;1-Y  
Material Identity Number: O697-97005  
Language: English Document Type: Journal Paper (JP)  
Treatment: Practical (P)

Abstract: The authors discuss **multithreading** issues by reviewing MFC's encapsulation of the Windows synchronization primitives. MFC provides classes to wrap the four main primitives: **mutexes**, semaphores, events and critical sections. The authors review the role of each of these objects in **multithreaded** programming, and look in detail at how MFC implements the wrappers. (0 Refs)

Subfile: C  
Copyright 1997, IEE

16/7/9 (Item 9 from file: 2)  
DIALOG(R)File 2:INSPEC  
(c) 2004 Institution of Electrical Engineers. All rts. reserv.

4553876 INSPEC Abstract Number: C9401-6150J-042

**Title: The GASO threads package an implementation of POSIX threads for VM/CMS**

Author(s): Gmach, R.  
Conference Title: Proceedings SHARE Europe Anniversary Meeting p. 265-82  
Publisher: SHARE Eur. Assoc, Geneva, Switzerland  
Publication Date: 1992 Country of Publication: Switzerland 752 pp.  
Conference Date: 28 Sept.-2 Oct. 1992 Conference Location: Davos, Switzerland  
Language: English Document Type: Conference Paper (PA)  
Treatment: Practical (P)

Abstract: Threads are a kind of lightweight processes, sharing the same address space. GASO is a user-level threads package for the IBM VM/SP operating system based on the Pthreads' interface specified by POSIX in their 1003.4A draft. This threads package is an easy-to-use library, which enables the programmer to write application in C, which are running in the normal CMS environment with **multiple threads** executing in parallel. It is possible for instance to have one thread waiting for user keyboard input, while all other threads keep performing any computations. Threads can be useful in structuring a program. Threads are also useful for having multiple copies of the same code running simultaneously; for example when a server program executes requests for clients, a thread can be used to execute each client's request. Threads communicate through shared variables-one thread may set a variable that another thread will read later. Provision for synchronization is implemented by means of **mutexes** and condition variables. (0 Refs)

Subfile: C

16/7/10 (Item 10 from file: 2)  
DIALOG(R)File 2:INSPEC  
(c) 2004 Institution of Electrical Engineers. All rts. reserv.

03142912 INSPEC Abstract Number: C88035591

**Title: VAX/SMP (VMS symmetric multiprocessing design)**

Author(s): Morse, K.D.; Gamache, R.N.

Author Affiliation: Digital Equipment Corp., Nashua, NH, USA

Journal: DEC Professional vol.7, no.4 p.64-74

Publication Date: April 1988 Country of Publication: USA

CODEN: DECPDJ ISSN: 0744-9216

Language: English Document Type: Journal Paper (JP)

Treatment: Practical (P); Product Review (R)

Abstract: The authors discuss the use of key synchronisation mechanisms (spinlocks) in the VMS/SMP design. They describe the development of SMP, how spinlocks came about and concepts in spinlock design. Other subjects covered are: granularity of spinlocks and device locks; IPL (interrupt priority level) usage; static spinlocks; and I/O subsystem synchronisation.  
(0 Refs)

Subfile: C

16/7/11 (Item 1 from file: 6)  
DIALOG(R)File 6:NTIS  
(c) 2004 NTIS, Intl Cpyrght All Rights Res. All rts. reserv.

1940979 NTIS Accession Number: PB96-148473

**NQTHM Mechanization of 'An Exercise in the Verification of Multi - Process Programs'**

Nagayama, M. ; Talcott, C.

Stanford Univ., CA. Dept. of Computer Science.

Corp. Source Codes: 009225004

Sponsor: Defense Advanced Research Projects Agency, Arlington, VA.;  
National Science Foundation, Washington, DC.

Report No.: STAN-CS-91-1370

Jun 91 86p

Languages: English

Journal Announcement: GRAI9610

Sponsored by Defense Advanced Research Projects Agency, Arlington, VA.  
and National Science Foundation, Washington, DC.

Order this product from NTIS by: phone at 1-800-553-NTIS (U.S. customers); (703)605-6000 (other countries); fax at (703)321-8547; and email at orders@ntis.fedworld.gov. NTIS is located at 5285 Port Royal Road, Springfield, VA, 22161, USA.

NTIS Prices: PC A05/MF A01

Country of Publication: United States

Contract No.: DARPA-N00039-84-C-0211; NSF-CCR-8718605

This report presents a formal verification of the local correctness of a **mutex** algorithm using the Boyer-Moore theorem prover. The formalization follows closely an informal proof of Manna and Pnueli. The proof method of Manna and Pnueli is to first extract from the program a set of states and induced transition system. One then proves suitable invariants. There are two variants of the proof. In the first (atomic) variant, compound tests involving quantification over a finite set are viewed as atomic operations. In the second (molecular) variant, this assumption is removed, making the details of the transition and proof somewhat more complicated. The original Manna-Pnueli proof was formulated in terms of finite sets. This led to a concise and elegant informal proof, however one that is not easy to mechanize in the Boyer-Moore logic. In the mechanized version the authors



use a dual isomorphic representation of program states based on finite sequences. The authors' approach was to outline the formal proof of each invariant, making explicit the case analyses, assumptions and properties of operations used. The outline served as the authors' guide in developing the formal proof. The resulting sequence of events follows the informal plan quite closely. The main difficulties encountered were in discovering the precise form of the lemmas and hints necessary to guide the theorem prover.

**16/7/12 (Item 1 from file: 8)**

DIALOG(R)File 8:EI Compendex(R)

(c) 2004 Elsevier Eng. Info. Inc. All rts. reserv.

03064976 E.I. Monthly No: EIM9105-021296

**Title: Simulation environment for n-process mutual exclusion algorithms.**

Author: Saeed, Faisel; Samadzadeh, M. H.; George, K. M.

Corporate Source: Oklahoma State Univ, OK, USA

Conference Title: Proceedings of the Twenty-First Annual Pittsburgh Conference Part 1 (of 5)

Conference Location: Pittsburgh, PA, USA Conference Date: 19900503

Sponsor: Univ of Pittsburgh, Dep of Electrical Engineering; IEEE, Pittsburgh Section; ISA; IEEE Systems, Man & Cybernetics Soc; Soc for Computer Simulation; et al

E.I. Conference No.: 14260

Source: Economics, Geography, Regional Science, Mathematics and Statistics Modeling and Simulation, Proceedings of the Annual Pittsburgh Conference v 21 pt part 1. Publ by Soc for Computer Simulation Int, San Diego, CA, USA. p 371-375

Publication Year: 1990

CODEN: MSPCD4

Language: English

Document Type: PA; (Conference Paper) Treatment: A; (Applications); T; (Theoretical)

Journal Announcement: 9105

Abstract: Mutual exclusion has been a challenging problem for computer scientists for several decades and a number of solutions have been suggested. This paper presents a scheme to define a simulation environment (written in Ada) to serve as a testing tool for the existing and future solutions to the mutual exclusion problem. The environment is partially validated by using classical correct and incorrect solutions to the mutual exclusion problem. (Author abstract) 25 Refs.

**16/7/13 (Item 1 from file: 95)**

DIALOG(R)File 95:TEME-Technology & Management

(c) 2004 FIZ TECHNIK. All rts. reserv.

01180982 E98020025046

**Echtzeitloesungen fuer Windows NT**

Wollert, JF

Preussag Noell Stahl- und Maschinenbau, Wuerzburg, D

Echtzeit 97, Kongress & Messe f. industrielle Busse und Netze, iNet 97,

Kongress und Messe f. industrielle Computer-Anwendungen, Kongressband, Conf. Proc., Wiesbaden, D, 9.-11. Sep, 19971997

Document type: Conference paper Language: German

Record type: Abstract

**ABSTRACT:**

Windows NT ist kein Echtzeitbetriebssystem fuer harte Echtzeitanforderung. Das ist offensichtlich. Im besonderen spricht die unpriorisierte Abarbeitung von **Mutex** -Tasks ueber FiFo-Buffer im Ring-0 des

Betriebssystemkerns gegen harte Echtzeit. Darüber hinaus koennen die dem Benutzer zugaenglichen Prioritaeten nie eine Task hoeher priorisieren als das Windowssystem. Auch die fehlende Prioritaetenvererbung bei Tasks der Klasse Echtzeit zur Vermeidung von Prioritaeteninversion und die unglueckliche Priorisierung von Treibern auf Kernel-Ebene sprechen gegen einen Einsatz von Windows NT als Echtzeitbetriebssystem. Dennoch ist Windows NT, basierend auf Microsofts Microkernel-Architektur, nicht nur eine schnelle Antwortzeit zu bescheinigen, sondern auch ein hoher Reifegrad. Gerade Anwendungen, die eine hohe Integrationstiefe mit Standardkomponenten haben und die eine ausgefeilte Benutzeroberflaeche benoetigen und gleichzeitig keine harten Echtzeitanforderungen erfuellen, koennen durch den Einsatz des Betriebssystems Windows NT produktiv realisiert werden. Hierzu tragen nicht zuletzt die vielfaeltigen und hoch entwickelten Entwicklungsumgebungen bis zum Kernel-Debugger bei. Reichen die Eigenschaften des originalen Windows NT nicht aus, so gibt es eine Reihe von kommerziell verfuegbaren Erweiterungen, die fast alle speziellen Probleme der Automatisierungstechnik abdecken. Von Echtzeitkernen ueber Kerneltreiber bis hin zu Multiprozessorapplikationen und Zusatzhardware sind nahezu alle Aufgabenfelder durch kommerzielle Anbieter abgedeckt. Bei der Betrachtung zukuenftiger Entwicklungen im Betriebssystemsektor und des verfuegbaren Programmierer Know-hows kommt man an der Ueberlegung 'Windows NT einsetzen oder nicht' nicht vorbei. Nicht zuletzt durch das Engagement von Microsoft in den industriellen Sektor, verbunden mit dem Trend zum IndustriePC, erscheint der Einsatz von Windows NT bis in den Prozessbereich - trotz der aufgefuehrten Einschraenkungen - ein Schritt in die richtige Richtung.

16/7/14 (Item 2 from file: 95)  
DIALOG(R)File 95:TEME-Technology & Management  
(c) 2004 FIZ TECHNIK. All rts. reserv.

01042880 E96110105258

**Asynchronous implementation of the SCPP-A counterflow pipeline processor**  
(Die asynchrone Implementierung des SCPP-A (Sproull Counterflow Pipeline Processor))

Koerver, WHFJ; Nedelchev, IM  
Univ. of Surrey, Guildford, GB; Synopsys Inc., Mountain View, USA  
IEE Proceedings - Computers and Digital Techniques, v143, n5, pp287-294, 1996

Document type: journal article Language: English  
Record type: Abstract  
ISSN: 1350-2387

**ABSTRACT:**

The SCPP-A design problem posed by C.E. Molnar and H. Schols describes an abstraction of the Sproull counterflow pipeline processor which mainly concentrates on pipeline control. Although this problem can be specified concisely, it raises design issues that are intrinsically difficult to deal with. An asynchronous implementation of SCPP-A is presented, where the speed of the resulting circuitry is regarded as the main design criterion. It describes the complete design path, from the formulation of formal functional specifications of the high-level components down to the implementation of the basic components. A number of additional aspects of the problem and the design are discussed, including design alternatives at various abstraction levels and a performance evaluation of the resulting circuitry. Altogether, this offers a good illustration of asynchronous design concerns.

16/7/15 (Item 3 from file: 95)

DIALOG(R)File 95:TEME-Technology & Management  
(c) 2004 FIZ TECHNIK. All rts. reserv.

00760515 E94030177232

**Keine Angst vor Threads. Die Multi - Thread -Architektur unter SunOS**  
(SunOS Multithread Architecture)

Schlede, F-M

UNIX open, v368, n3, pp46-51, 1994

Document type: journal article Language: German

Record type: Abstract

ISSN: 0943-8416

**ABSTRACT:**

Im Artikel werden am Beispiel von Solaris Threads und ihre Programmierung erlaeutert. Das Programmiermodell fuer die Threads in SunOS basiert auf zwei Ebenen: Einmal werden im Thread-Interface das Programmiermodell aus der Sicht des Anwendungsprogrammierers festgelegt und andererseits gibt es die 'Lightweight Processes' (LWP); Dienste, die vom Betriebssystem bereitgestellt werden muessen. Allgemeine Thread-Operationen und Aufbau der **Multi - Thread -Architektur** werden umfassend beschrieben. Vier verschiedene Hilfsmittel fuer die Synchronisation von Threads werden erlaeutert: Mutual Exclusion ( **Mutex** ) Locks, Counting Semaphores, Condition Variables und Multiple Reader. Ein weiterer Ausblick zu Einsatzmoeglichkeiten der **Multi - Threaded** Programmierung wird gegeben.

**16/7/16 (Item 1 from file: 233)**

DIALOG(R)File 233:Internet & Personal Comp. Abs.

(c) 2003 EBSCO Pub. All rts. reserv.

00543477 99WJ08-002

**Creating Win32 lock hierarchies**

Claar, Jeff

Windows Developer's Journal , August 1, 1999 , v10 n8 p24-32, 7 Page(s)

ISSN: 1083-9887

Describes a solution for Win32's lack of any support for a lock hierarchy, wherein a **multithreaded** application developer specifies that order in which **mutexes** must be locked, preventing deadlocks from occurring. Notes that for a deadlock to occur, a **multithreaded** app must have at least two synchronization objects and two threads in the program must acquire at least two of the synchronization objects in a different order. Presents a C++ class for lock hierarchies and explains that whenever a critical section or **mutex** is entered, the object will perform a check to determine if the lock hierarchy has been violated and break the program into the debugger if it has. Indicates that when a thread acquires a lock, this program verifies that the acquisition does not violate the lock hierarchy and it acquires the lock. Also covers the program's behavior when the thread releases a lock. Includes three program listings and one illustration. (jon)

**16/7/17 (Item 2 from file: 233)**

DIALOG(R)File 233:Internet & Personal Comp. Abs.

(c) 2003 EBSCO Pub. All rts. reserv.

00525870 99CQ02-002

**Adding Level-2 thread safety to existing objects -- The code required to share an object among multiple threads is tedious and error prone. But it can be neatly encapsulated**

Richards, Etienne

C/C++ Users Journal , February 1, 1999 , v17 n2 p61-71, 7 Page(s)  
ISSN: 1075-2838

Describes a solution for implementing Level-2 thread safety in code, where only one thread is allowed to operate on an object at one time, which has a similar effect to the Java synchronized keyword. States that this solution can be used with most existing classes without changing the implementation of the class that required synchronized access. Claims that with this solution a programmer can use most Level-1 thread-safe objects (TSOs) as Level-2 TSOs; no code change is needed for existing objects; client code requires minimal code change; and the programmer can change safety policy without changing the object being protected. Explains that the design is based on the concept of a monitor type of object, which is implemented by using **mutex** semaphores, where mutual exclusion objects are used to restrict access to a shared resource. Attention is also given to the stack-based shield class. Includes six program listings and one reference.

16/7/18 (Item 3 from file: 233)

DIALOG(R)File 233:Internet & Personal Comp. Abs.  
(c) 2003 EBSCO Pub. All rts. reserv.

00484129 98DD01-007

**Mutual exclusion and synchronization in Java -- Mimicking Win32 synchronization mechanisms**

Ford, Don

Dr. Dobb's Journal , January 1, 1998 , v23 n1 p62-75, 9 Page(s)  
ISSN: 1044-789X

Discusses the concurrency features of Java, the tools for serializing access to shared resources and synchronizing separate threads of execution, and Java classes for mutual exclusion and synchronization that mimic the behavior and interfaces of Win32 synchronization mechanisms. Notes that the classes were originally developed for applications that act as Java servers for multiple Java applets used by a network management system. Describes well-behaved threads, thread priorities, a special mutual-exclusion class, **mutexes** , critical sections, events, and semaphores. Concludes that appropriate use of **multithreading** , combined with the proper tools and techniques, help create powerful Java applications and applets. Includes two diagrams. (dpm)

16/7/19 (Item 4 from file: 233)

DIALOG(R)File 233:Internet & Personal Comp. Abs.  
(c) 2003 EBSCO Pub. All rts. reserv.

00435525 96PI09-075

**Prevent multiple Win32 application instances -- How to detect whether a copy of your program is already running**

Proise, Jeff

PC Magazine , September 10, 1996 , v15 n15 p319-330, 5 Page(s)  
ISSN: 0888-8507

Instructs the reader on how to set up an application that will count the number of instances that a 32-bit application is running at once. Explains that the hPrevInstance query in Windows 3.x is ineffective due to a difference in a way multiple applications are handled in the 32-bit Windows 95 and Windows NT. However, points out that several variations of the technique are available that will count the number of times a program is running concurrently. Spotlights the FindWindows method, the WinMain function using a global atom table, named **mutexes** , and a shared memory instance count. Notes that there are other methods also available. Includes

four program listings. (kgh)

**16/7/20 (Item 5 from file: 233)**

DIALOG(R)File 233:Internet & Personal Comp. Abs.

(c) 2003 EBSCO Pub. All rts. reserv.

00402131 95BY11-021

**Windows NT threads: to truly reap the rewards of a multiprocessor NT system, you have to use threads**

Prasad, Shashi

BYTE , November 1, 1995 , v20 n11 p253-254, 2 Page(s)

ISSN: 0360-5280

Focuses on the Win32 interface in Windows NT for developing **multithreading** applications, which offers one of the best choices for harnessing the power of symmetric **multiprocessing** machines. Explains that an NT process has its own virtual address space and owns system resources; and threads in NT have 32 different priority levels, and the dispatcher module, which is responsible for thread-scheduling, uses a preemptive priority scheduler. Notes that in a **multithreaded** program, all threads within the program run in a single address space, and in NT, **mutexes** are used to serialize critical sections of code, that is, segments of code in which a thread accesses shared, modifiable data, and where state changes happen over several instructions. Considers the performance and scalability of threaded applications; and notes that I/O-completion ports are designed to handle asynchronous or overlapped I/O. Includes one code fragment. (jo)

**16/7/21 (Item 6 from file: 233)**

DIALOG(R)File 233:Internet & Personal Comp. Abs.

(c) 2003 EBSCO Pub. All rts. reserv.

00390911 95LA07-021

**The effects of pre-emption -- You can avoid OS crashes by 'tying down' shared resources**

Day, Michael

LAN Times , July 3, 1995 , v12 n13 p93-94, 2 Page(s)

ISSN: 1040-5917

INSIDE OSES column discusses a solution for operating system crashes in a preemptive environment. Explains how to tie down resources which are shared among thread or processes, including memory, files, pipes, devices, and other resources. Offers several examples, including an algorithm when programming for a non-preemptive **multitasker** such as NetWare. Also discusses semaphores, **mutexes** , locks, and handles, which are tools provided by preemptive OSes to allow for sharing resources by processes that may be preempted between instructions. Includes two diagrams. (CH)

**16/7/22 (Item 7 from file: 233)**

DIALOG(R)File 233:Internet & Personal Comp. Abs.

(c) 2003 EBSCO Pub. All rts. reserv.

00373401 95UR01-002

**Put multiprocessing systems to work, II -- Programming for multiprocessors requires the use of unusual features such as spin locks, mutex locks, barrier synchronization, and ...**

Walter, Stephen

UNIX Review , January 1, 1995 , v13 n1 p39-47, 7 Page(s)

ISSN: 0742-3136

The second in a series of articles discusses multiprocessor system

programming. Details Amadahl's Law and its implications in code parallelization. Describes the workpile algorithm and its limitations. Provides insights into how programmers can maximize parallel performance in CPU-bound applications. Differentiates between cooperative and client-server threads. Presents eight POSIX pthreads functions. Defines spin locks and barrier synchron Features approaches to the solution matrix multiplication proble using cooperative threads to implement parallelism. Illustrate client-server threading through a resource management problem. (ACD)

16/7/23 (Item 8 from file: 233)  
DIALOG(R)File 233:Internet & Personal Comp. Abs.  
(c) 2003 EBSCO Pub. All rts. reserv.

00324829 93PI09-286

**Installing traffic lights under Windows NT**

Petzold, Charles

PC Magazine , September 28, 1993 , v12 n16 p339-344, 6 Page(s)

ISSN: 0888-8507

ENVIRONMENTS column analyzes the programming equivalent of a traffic light to help coordinate and synchronize the thread traffic for a **multithreaded** operating system such as Windows NT. Utilizes a simplified example of two threads sharing a global array of five integers. Presents four C language program listings for Windows NT on IBM PC compatibles, which illustrate: an unsynchronized version of an update and display program; how to use critical sections (blocks of code that should not be interrupted) to avoid thread collisions; how to use **mutex** (mutual exclusion) objects to avoid thread collisions; and how to use event objects to fully synchronize the threads. Covers the factors involved in Windows NT switching control between the two threads; functions for using critical sections; how a thread requests ownership of a **mutex** object; and how Windows NT event objects work. Listings available free on PCMagNet. Includes eight program listings. (jo)

16/7/24 (Item 9 from file: 233)  
DIALOG(R)File 233:Internet & Personal Comp. Abs.  
(c) 2003 EBSCO Pub. All rts. reserv.

00243208 91BY06-038

**Symmetry, thy name is Unix Unix SVR4/MP: a new standard for multiprocessing with Unix. How does it fit into the open-systems picture?**

Nudelman, Mark

BYTE , June 1, 1991 , v16 n6 p245-253, 7 Pages

ISSN: 0360-5280

Discusses symmetric **multiprocessing** (SMP) under Unix, addressing the difficulties in migrating uniprocessor standards to multiprocessor environments. Describes compatibility and porting considerations in designing architectural specifications of SMP systems based on AT&T Unix System V release 4.0 (SVR4/MP). Explains scalability goals of SVR4/MP, and target hardware requirements to support automatic coherency among all system memory caches as well as to support interprocessor interrupt capability. Discusses efforts in **multithreading** SVR4 to implement a mutual-exclusion ( **mutex** ) lock associated with each of the kernel's data structures, as well as recursion and automatic release on sleep features of **mutex** locks. Contains one illustration. (jo)

?

25/7/1 (Item 1 from file: 2)

DIALOG(R)File 2:INSPEC

(c) 2004 Institution of Electrical Engineers. All rts. reserv.

03856709 INSPEC Abstract Number: B91026686, C91031645

**Title: TOP/PDT: a toolkit for the development of communication protocols**

Author(s): Albertengo, G.; Forno, S.; Fumagalli, A.

Author Affiliation: Dept. of Electron., Politecnico di Torino, Italy

Journal: IEEE Journal on Selected Areas in Communications vol.8, no.9  
p.1763-70

Publication Date: Dec. 1990 Country of Publication: USA

CODEN: ISACEM ISSN: 0733-8716

U.S. Copyright Clearance Center Code: 0733-8716/90/1200-1763\$01.00

Language: English Document Type: Journal Paper (JP)

Treatment: Applications (A); Practical (P)

Abstract: TOP/PDT, a toolkit for the development and the simulation of communication protocols, is presented. A protocol is described by a high-level language, derived from SDL, and named PDL, short for protocol description language. A compiler translates PDL programs into C programs; these are then compiled and linked using the utilities of the host operating system. TOP/PDT is an open environment: to add functionalities or to optimize a protocol, users can add their **own** functions, usually written in C, to the existing function library. The validation of the protocol is done by simulation. PDL allows the characterization of the data link(s) used to transmit and provides a set of measurement functions to evaluate the protocol performance. The modular structure of TOP/PDT and the use of the C programming language make it easily portable. The executable program generated using TOP/PDT can also run without a host operating system, since the PDT kernel provides a minimal operating system able to handle an unlimited **number** of **processes** and ensures their **mutual exclusion** in the access to shared data. (23 Refs)

Subfile: B C

25/7/2 (Item 2 from file: 2)

DIALOG(R)File 2:INSPEC

(c) 2004 Institution of Electrical Engineers. All rts. reserv.

03820182 INSPEC Abstract Number: C91016528

**Title: Mutual exclusion in arbitrary networks when process identity numbers are not distinct**

Author(s): Bouabdallah, A.; Naimi, M.

Author Affiliation: Lab. d'Inf., Besancon Univ., France

Conference Title: Ninth Annual International Phoenix Conference on Computers and Communications (Cat. No.90CH2799-5) p.885-6

Publisher: IEEE Comput. Soc. Press, Los Alamitos, CA, USA

Publication Date: 1990 Country of Publication: USA xxii+910 pp.

ISBN: 0 8186 2030 7

U.S. Copyright Clearance Center Code: CH2799-5/90/0000-0885\$01.00

Conference Sponsor: IEEE; Arizona State Univ.; Univ. Arizona

Conference Date: 21-23 March 1990 Conference Location: Scottsdale, AZ, USA

Language: English Document Type: Conference Paper (PA)

Treatment: Practical (P)

Abstract: A study of the problem of the **mutual exclusion** of critical sections of **several** distributed **processes** is presented. Each process is given with a particular section of code called its critical section. The problem is to design protocol to ensure that, at a given time, no more than one process is executing its **own** critical section. A distributed

algorithm for **mutual exclusion** working in arbitrary networks is proposed. This algorithm does not use the logical clock, and the identities of nodes are not necessarily distinct. Initially, each process knows only its adjacent channels and knows them by local identities. The network is organized as a logical rooted tree where the root holds the token. When a process wants to enter the critical section, it sends a request message over an outgoing channel toward the root and waits for the token. The number of messages required for every request is bounded by  $2d$  where  $d$  is the diameter of the network. (3 Refs)

Subfile: C

25/7/3 (Item 3 from file: 2)

DIALOG(R)File 2:INSPEC

(c) 2004 Institution of Electrical Engineers. All rts. reserv.

03390614 INSPEC Abstract Number: C89038999

**Title: A decentralized virtual memory scheme implemented on an emulated multiprocessor**

Author(s): Brorsson, M.

Author Affiliation: Dept. of Comput. Eng., Lund Univ., Sweden

Conference Title: Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences. Vol.I: Architecture Track (IEEE Cat. No. 89TH0242-8) p.286-95 vol.1

Editor(s): Hoevel, L.W.; Milutinovic, V.

Publisher: IEEE Comput. Soc. Press, Washington, DC, USA

Publication Date: 1989 Country of Publication: USA viii+492 pp.

ISBN: 0 8186 1911 2

U.S. Copyright Clearance Center Code: 0073-1129/89/0000-0286\$01.00

Conference Sponsor: IEEE; Univ. Hawaii; PRIISM; ACM

Conference Date: 3-6 Jan. 1989 Conference Location: Kailua-Kona, HI, USA

Language: English Document Type: Conference Paper (PA)

Treatment: Practical (P)

**Abstract:** A decentralized scheme for virtual memory management on MIMD (multiple-instruction-multiple-data) multiprocessors with shared memory has been developed. Control and data structures are kept local to the processing elements (PE), which reduces the global traffic and makes a high degree of parallelism possible. Each of the PEs in the target architecture consists of a processor and part of the shared memory and is connected to the others by a common bus. The traditional approach, based on replication or sharing of data structures is not suitable in this case when the number of PEs is of the magnitude of 100. This is due to the excessive global traffic caused by consistency or **mutual exclusion** protocols. A variant of the Dennings working set page replacement algorithm is used, in which each process **owns** a page list. Shared pages are not present in more than one list, and it is shown that this will not increase the page fault rate in most cases. (11 Refs)

Subfile: C

25/7/4 (Item 4 from file: 2)

DIALOG(R)File 2:INSPEC

(c) 2004 Institution of Electrical Engineers. All rts. reserv.

02716429 INSPEC Abstract Number: C86044530

**Title: Implementation of an executive multitask real-time monitor on an IBM-PC extension card**

Author(s): Salembier, D.

Journal: Electronique Industrielle no.104 p.49-53



Publication Date: 15 March 1986 Country of Publication: France

CODEN: EIDUDA ISSN: 0244-903X

Language: French Document Type: Journal Paper (JP)

Treatment: Practical (P)

Abstract: The architecture of the P-Sceptus executive system developed by Elios Informatique is explained with definition of its kernel and its functions and services offered in management of tasks, intertask signalling and communication, **mutual exclusion** and task abortion. These simple services are called by a TRAP instruction so that the user can write his **own** application, in either assembly language or C. Programs for A/D conversion by the iSBX311 card are listed in BASIC and C for comparison. Suggested applications include management of programmable controllers, simultaneous data acquisition and analysis, production line automation, telecommunication traffic or response time measurement, robot control and industrial process monitoring. (0 Refs)

Subfile: C

25/7/5 (Item 5 from file: 2)

DIALOG(R) File 2:INSPEC

(c) 2004 Institution of Electrical Engineers. All rts. reserv.

02405790 INSPEC Abstract Number: C85014347

**Title: Symmetric and economical solutions to the mutual exclusion problem in a distributed system**

Author(s): Cohen, S.; Lehmann, D.; Pnueli, A.

Author Affiliation: Inst. of Maths. & Comput. Sci., Hebrew Univ., Jerusalem, Israel

Journal: Theoretical Computer Science vol.34, no.1-2 p.215-25

Publication Date: Nov. 1984 Country of Publication: Netherlands

CODEN: TCSCDI ISSN: 0304-3975

U.S. Copyright Clearance Center Code: 0304-3975/84/\$3.00

Language: English Document Type: Journal Paper (JP)

Treatment: Theoretical (T)

Abstract: The **mutual exclusion** problem in a distributed system, in which each process has a memory of its **own**, into which it has exclusive write privileges but from which others may read, is reconsidered. Symmetric solutions are looked for. It is shown that, though no such solution may be deterministic, there are probabilistic solutions. Various solutions are provided for two processes, and then a solution is proposed for any **number** of **processes**. The solutions offered are amenable to a formal proof of their correctness with a small effort. The solutions are correct even against a very well informed schedule, unlike Rabin's probabilistic solution to the **mutual exclusion** problem in a centralised system. Some of the solutions are correct even against an evil scheduler which knows in advance the results of the future random draws, in sharp contrast with the algorithms of Lehmann and Rabin (1981). The solutions are economical; **mutual exclusion** between two processes may be achieved with variables capable of holding four different values (to be compared with Peterson and Fischer's three), and **mutual exclusion** between n processes may be achieved with variables capable of holding ten different values (to be compared with Peterson and Fischer's fourteen). All solutions have been attained by careful reasoning and not by an exhaustive computer search; they exhibit general principles of design which may be useful in solving other similar problems. (15 Refs)

Subfile: C

25/7/6 (Item 1 from file: 8)

DIALOG(R) File 8:Ei Compendex(R)

(c) 2004 Elsevier Eng. Info. Inc. All rts. reserv.

04295482 E.I. No: EIP95122938383

**Title: Recoverable user-level mutual exclusion**

Author: Bohannon, Philip; Lieuwen, Daniel; Silberschatz, Avi; Sudarshan, S.; Gava, Jacques

Corporate Source: AT&T Bell Lab, Murray Hill, NJ, USA

Conference Title: Proceedings of the 1995 7th IEEE Symposium on Parallel and Distributed Processing

Conference Location: San Antonio, TX, USA Conference Date: 19951025-19951028

Sponsor: IEEE

E.I. Conference No.: 44002

Source: IEEE Symposium on Parallel and Distributed Processing - Proceedings 1995. IEEE, Los Alamitos, CA, USA, 95TB8131. p 293-301


Publication Year: 1995

CODEN: PSPDF8 ISSN: 1063-6374

Language: English

Document Type: CA; (Conference Article) Treatment: A; (Applications)

Journal Announcement: 9601W4

 Abstract: **Mutual exclusion** primitives based on user-level atomic instructions (often called spin locks) have proven to be much more efficient than operating-system semaphores in situations where the contention on the semaphore is low. However, many of these spin lock schemes do not permit registration of **ownership** to be carried out atomically with acquisition, potentially leaving the **ownership** undetermined if a process dies (or makes very slow progress) at a critical point in the registration code. We present an algorithm which can ensure the successful registration of **ownership** of a spin lock, regardless of where processes fail. Thus, our spin lock implementation is 'recoverable'. The determination of a spin lock's **ownership** can potentially be used to restore resources protected by the spin lock to consistency and then release the spin lock. Other processes using the lock can then continue to function normally, improving fault resiliency for the application. Our algorithm provides very fast lock acquisition when the acquisition is uncontested (comparable in speed to a simple test-and-set based spin lock), and we prove it works even on the weak memory consistency models implemented by many modern multiprocessor computer systems. (Author abstract) 14 Refs.

25/7/7 (Item 1 from file: 34)

DIALOG(R) File 34:SciSearch(R) Cited Ref Sci

(c) 2004 Inst for Sci Info. All rts. reserv.

05094796 Genuine Article#: TQ208 Number of References: 20

**Title: TIME CONTENTION TRADE-OFFS FOR MULTIPROCESSOR SYNCHRONIZATION**

Author(s): ANDERSON JH; YANG JH

Corporate Source: UNIV N CAROLINA, DEPT COMP SCI/CHAPEL HILL//NC/27599; UNIV MARYLAND, DEPT COMP SCI/COLLEGE PK//MD/20742

Journal: INFORMATION AND COMPUTATION, 1996, V124, N1 (JAN 10), P68-84

ISSN: 0890-5401

Language: ENGLISH Document Type: ARTICLE

Abstract: We establish trade-offs between time complexity and write-and access-contention for solutions to the **mutual exclusion** problem. The write-contention (access-contention) of a concurrent program is the **number of processes** that may be simultaneously enabled to write (access by reading and/or writing) the same shared variable. Our notion of time complexity distinguishes between local and remote accesses of shared memory. We show that, for any N-process **mutual exclusion**

algorithm, if write-contention is  $w$ , and if at most  $v$  remote variables can be accessed by a single atomic operation, then there exists an execution involving only one process in which that process executes  $\Omega(\log(vw)N)$  remote operations for entry into its critical section. We further show that, among these operations,  $\Omega(\log(vw)N)$  distinct remote variables are accessed. For algorithms with access-contention  $c$ , we show that the latter bound can be improved to  $\Omega(\log(vc)N)$ . The last two of these bounds imply that a trade-off between contention and time complexity exists even if coherent caching techniques are employed. In most shared-memory multiprocessors, an atomic operation may access only a constant number of remote variables. In fact, most commonly-available synchronization primitives (e.g., read, write, test-and-set, load-and-store, compare-and-swap, and fetch-and-add) access only one remote variable. In this case, the first and the last of our bounds are asymptotically tight. Our results have a number of important implications regarding specific concurrent programming problems. For example, the time bounds that we establish apply not only to the **mutual exclusion** problem, but also to a class of decision problems that includes the leader-election problem. Also, because the execution that establishes these bounds involves only one process, it follows that '**fast mutual exclusion**' requires arbitrarily high write-contention. Although such conclusions are interesting in their **own** right, we believe that the most important contribution of our work is to identify a time complexity measure for asynchronous concurrent programs that strikes a balance between being conceptually simple and having a tangible connection to real performance. (C) 1996 Academic Press, Inc.

25/7/8 (Item 2 from file: 34)

DIALOG(R)File 34:SciSearch(R) Cited Ref Sci  
(c) 2004 Inst for Sci Info. All rts. reserv.

03325572 Genuine Article#: NY211 Number of References: 172

**Title: THE INTERDISCIPLINARY STUDY OF COORDINATION**

Author(s): MALONE TW; CROWSTON K

Corporate Source: MIT,CTR COORDINATED SCI/CAMBRIDGE//MA/02139; UNIV MICHIGAN,SCH BUSINESS ADM/ANN ARBOR//MI/48109

Journal: ACM COMPUTING SURVEYS, 1994, V26, N1 (MAR), P87-119

ISSN: 0360-0300

Language: ENGLISH Document Type: REVIEW

**Abstract:** This survey characterizes an emerging research area, sometimes called coordination theory, that focuses on the interdisciplinary study of coordination. Research in this area uses and extends ideas about coordination from disciplines such as computer science, organization theory, operations research, economics, linguistics, and psychology.

A key insight of the framework presented here is that coordination can be seen as the process of managing dependencies among activities. Further progress, therefore, should be possible by characterizing different kinds of dependencies and identifying the coordination processes that can be used to manage them. A **variety of processes** are analyzed from this perspective, and commonalities across disciplines are identified. Processes analyzed include those for managing shared resources, producer/consumer relationships, simultaneity constraints, and task/subtask dependencies.

Section 3 summarizes ways of applying a coordination perspective in three different domains: (1) understanding the effects of information technology on human organizations and markets, (2) designing cooperative work tools, and (3) designing distributed and parallel

computer systems. In the final section, elements of a research agenda in this new area are briefly outlined.

25/7/9 (Item 1 from file: 35)

DIALOG(R) File 35:Dissertation Abs Online  
(c) 2004 ProQuest Info&Learning. All rts. reserv.

1031526 ORDER NO: AAD88-25074

**OPERATING SYSTEM DATA STRUCTURES FOR SHARED-MEMORY MIMD MACHINES WITH  
FETCH-AND-ADD**

Author: WILSON, JAMES M.

Degree: PH.D.

Year: 1988

Corporate Source/Institution: NEW YORK UNIVERSITY (0146)

ADVISER: ALLAN GOTTLIEB

Source: VOLUME 49/09-B OF DISSERTATION ABSTRACTS INTERNATIONAL.

PAGE 3867. 497 PAGES

Ideally, procedures and data structures on a shared-memory MIMD machine should be serialization-free and concurrently accessible to avoid (potential) performance-limiting bottlenecks. The fetch-and-add coordination primitive, in conjunction with "combining" interconnection networks, has been proposed as a means for achieving this goal. The first is essentially an indivisible add-to-memory and the second combines simultaneous requests to the same memory location.

In this thesis we address serialization-free memory and process management for a shared-memory MIMD machine with fetch-and-add and a combining network. To meet this goal we adopt a self-service paradigm for the operating system that permits each processing element (PE) to service its **own** requests (thereby avoiding central server bottlenecks). The success of this approach depends upon the use of concurrently accessible data structures to hold data shared among the PEs.

We begin by reviewing existing fetch-and-add based queue and multiqueue (a compressed queue) implementations that support concurrent queue insertion and deletion. We then extend these implementations to include a new operations (e.g., the removal of an interior queue item) and new data structure representations (e.g., linked lists).

Parallel memory allocation algorithms, many based on the modified queue and multiqueue data structures, are then given. These algorithms include parallel analogs to a number of existing serial algorithms such as Knuth's boundary tag method and the binary buddy system.

Next, we define a set of primitives that permit **various task** activities, such as creation and scheduling, to be done in parallel. Task-switching readers/writers and event primitives are given as well. In the readers/writers implementations, reader activity is fully parallel in the absence of writers. An important feature of both the readers/writers and event implementations is that tasks waiting for a resource can be resumed in parallel by multiple PEs.

We then demonstrate how high-level parallel programming constructs (e.g., parallel loops) may be implemented via the task primitives and the queue and multiqueue data structures.

Finally, we prove that one of the readers/writers implementations satisfies certain correctness criteria including freedom from deadlock and the **mutual exclusion** of readers and writers.

File 347:JAPIO Oct 1976-2003/Oct(Updated 040202)  
(c) 2004 JPO & JAPIO  
File 350:Derwent WPIX 1963-2004/UD,UM &UP=200409  
(c) 2004 Thomson Derwent  
File 348:EUROPEAN PATENTS 1978-2004/Jan W05  
(c) 2004 European Patent Office  
File 349:PCT FULLTEXT 1979-2002/UB=20040205,UT=20040129  
(c) 2004 WIPO/Univentio

Set	Items	Description
S1	466	AU='WAGNER M'
S2	23	AU='WAGNER M P'
S3	98	AU='WAGNER MICHAEL'
S4	14	AU='WAGNER MICHAEL DR':AU='WAGNER MICHAEL DR ING'
S5	335	MUTEX? OR MUTUAL()EXCUSION? OR MUT()EX
S6	600	S1:S4
S7	0	S6 AND S5

File 348:EUROPEAN PATENTS 1978-2004/Jan W05  
(c) 2004 European Patent Office  
File 350:Derwent WPIX 1963-2004/UD,UM &UP=200409  
(c) 2004 Thomson Derwent

Set	Items	Description
S1	130	MUTEX? OR MUTUAL()EXCUSION? OR MUT()EX OR MUTUALEXCLUSION?
S2	2533164	THREAD???? ? OR PROCESS OR PROCESSES OR PROCESSING OR TASK- ???? ?
S3	46091	(PLURALITY OR MANY OR MULTI OR SEVERAL OR NUMEROUS OR MULT- IPLE OR MULTITUD? OR MULTIPLICITY OR PLURIF? OR VARIOUS OR VA- RIETY) (1W)S2
S4	43083	(GROUP???? ? OR CLUSTER??? ? OR NUMBER OR NETWORK? ? OR CH- AIN? ? OR SERIES OR THREE OR COLLECTION? OR THIRD OR ASSORT? - OR RANGE? ?) (1W)S2
S5	3614	MULTITHREAD? OR MULTIPROCESS OR MULTIPROCESSES OR MULTIPRO- CESSING OR MULTITASK?
S6	0	GHOSTLOCK? OR GHOST()LOCK??? ?
S7	129294	OWN OR OWNS OR OWNER? OR OWNED
S8	993	OWNING
S9	836	CO()S7:S8 OR S7:S8(2N) (TEMPORAR??? ? OR BRIEF?? ? OR INTER- IM OR MOMENTAR? OR TRANSIT?)
S10	1401	S7:S8(2N) (PARTIAL? OR PORTION? OR PART OR DIVID?? ? OR DIV- ISION? OR SUBDIVID? OR SUBDIVISION? OR PARTITION? OR REDIVID? OR REDIVISION? OR SPLIT???? ?)
S11	31	PARTOWNER? OR COOWN?
S12	208	MUTUAL(1W)EXCLUSION?
S13	258	(PLURALITY OR MANY OR MULTI OR SEVERAL OR NUMEROUS OR MULT- IPLE OR MULTITUD? OR MULTIPLICITY OR PLURIF? OR VARIOUS OR VA- RIETY) (1W) (S7 OR OWNING)
S14	619	(GROUP???? ? OR CLUSTER??? ? OR NUMBER OR NETWORK? ? OR CH- AIN? ? OR SERIES OR THREE OR COLLECTION? OR THIRD OR ASSORT? - OR RANGE? ?) (1W)S7:S8
S15	43	(S1 OR S12) (25N)S3:S5
S16	0	S15(25N) (S6 OR S9:S11 OR S13:S14)
S17	4	S15 AND (S6 OR S9:S11 OR S13:S14)

17/5,K/1 (Item 1 from file: 348)  
DIALOG(R)File 348:EUROPEAN PATENTS  
(c) 2004 European Patent Office. All rts. reserv.

01106332

**Determinism in a multiprocessor computer system and monitor and processor therefor**

**Determinismus in einem Mehrprozessorrechnersystem und Monitor und Prozessor dafur**

**Determinisme dans un systeme d'ordinateur a multiprocesseurs et moniteur et processeur pour cela**

PATENT ASSIGNEE:

SUN MICROSYSTEMS, INC., (1392733), 901 San Antonio Road, Palo Alto,  
California 94303, (US), (Applicant designated States: all)

INVENTOR:

Williams, Emrys John, 1063 Morse Avenue Nr. 3-205, Sunnyvale, California  
94089, (US)

LEGAL REPRESENTATIVE:

Harris, Ian Richard (72231), D. Young & Co., 21 New Fetter Lane, London  
EC4A 1DA, (GB)

PATENT (CC, No, Kind, Date): EP 969374 A2 000105 (Basic)

APPLICATION (CC, No, Date): EP 99304904 990622;

PRIORITY (CC, No, Date): US 106883 980630

DESIGNATED STATES: AT; BE; CH; CY; DE; DK; ES; FI; FR; GB; GR; IE; IT; LI;

LU; MC; NL; PT; SE  
EXTENDED DESIGNATED STATES: AL; LT; LV; MK; RO; SI  
INTERNATIONAL PATENT CLASS: G06F-011/18; G06F-011/30

ABSTRACT EP 969374 A2

A multiprocessor computer system which provides fault tolerance includes a number of processing sets. At least one of the processing sets is operable asynchronously of a second processing set. A monitor is connected to receive I/O operations output from the processing sets for identifying faulty operation of those units. The monitor is also operable to synchronise operation of the processing sets by signalling the processing sets on receipt of outputs from those units indicative of a plurality of them being at an equivalent stage of processing. The monitor provides for buffering of I/O operations output from the processing sets and for selective forwarding of those I/O operations to an external I/O bus. The processing set may be formed from a single processor or from multiple processors.

ABSTRACT WORD COUNT: 127

NOTE:

Figure number on first page: 5

LEGAL STATUS (Type, Pub Date, Kind, Text):


Assignee: 030423 A2 Transfer of rights to new applicant: Sun  
Microsystems, Inc. (2616592) 4150 Network  
Circle Santa Clara, California 95054 US  
Application: 20000105 A2 Published application without search report  
Change: 031126 A2 International Patent Classification changed:  
20031010

LANGUAGE (Publication, Procedural, Application): English; English; English

FULLTEXT AVAILABILITY:

Available Text	Language	Update	Word Count
CLAIMS A	(English)	200001	1545
SPEC A	(English)	200001	14112
Total word count - document A			15657
Total word count - document B			0
Total word count - documents A + B			15657

...SPECIFICATION control is exercised over the way the multiple processors of a single processing set use **mutual exclusion** primitives ( **mutexes** ). In practice it is the **various processing threads** in the processors which use the **mutexes** . In an MP machine, to provide a reasonably simple programming environment, the processors (or rather...

 ...Proper use of mutexes makes the processors of an MP system each act on its **own portion** of the total real state, with the important restriction that other processors will not modify...

...the real state, the only variable left undetermined is the order of acquisition of the **mutexes** by the various processors. If the processors (processor threads) in the **various processing** sets acquire and release **mutexes** in the same order, then all the modifications to the real state are wholly determined...voted and synchronized cycles on the I/O bus and so will automatically provide equivalent **mutex** ordering on **multiple processing** sets. No additional monitor capabilities are needed.

Yet another approach for the monitor unit to control **mutex** ordering is to use a combination of the above approaches. A relatively small number of...

17/5,K/2 (Item 2 from file: 348)  
DIALOG(R)File 348:EUROPEAN PATENTS  
(c) 2004 European Patent Office. All rts. reserv.

01106330

**I/O handling for a fault tolerant multiprocessor computer system**

**E/A-Verarbeitung für ein fehlertolerantes Mehrrechnersystem**

**Traitement d'E/S pour un système d'ordinateur à multiprocesseurs tolerant des fautes**

PATENT ASSIGNEE:

Sun Microsystems, Inc., (2616592), 4150 Network Circle, Santa Clara,  
California 95054, (US), (Applicant designated States: all)

INVENTOR:

Williams, Emrys John, 1063 Morse Avenue Nr.3-205, Sunnyvale, California  
94089, (US)

LEGAL REPRESENTATIVE:

Harris, Ian Richard (72231), D. Young & Co., 21 New Fetter Lane, London  
EC4A 1DA, (GB)

PATENT (CC, No, Kind, Date): EP 969373 A2 000105 (Basic)  
EP 969373 A3 030521

APPLICATION (CC, No, Date): EP 99304896 990622;

PRIORITY (CC, No, Date): US 107973 980630

DESIGNATED STATES: AT; BE; CH; CY; DE; DK; ES; FI; FR; GB; GR; IE; IT; LI;  
LU; MC; NL; PT; SE

EXTENDED DESIGNATED STATES: AL; LT; LV; MK; RO; SI

INTERNATIONAL PATENT CLASS: G06F-011/18

ABSTRACT EP 969373 A2

A multiprocessor computer system which provides fault tolerance includes a number of processing sets. At least one of the processing sets is operable asynchronously of a second processing set. A monitor is connected to receive I/O operations output from the processing sets for identifying faulty operation of those units. The monitor is also operable to synchronize operation of the processing sets by signalling the processing sets on receipt of outputs from those units indicative of a plurality of them being at an equivalent stage of processing. The common monitor can be operable to determine not only faulty operation of the processing sets, but also to synchronize those units by monitoring the I/O operations output from the units. The monitor provides for buffering of I/O operations output from the processing sets and for selective forwarding of those I/O operations to an external I/O bus. A processing set may be formed from a single processor, or may be formed from multiple symmetric processors. A mutex ordering scheme can be provided to ensure deterministic access by the individual processors of the processing sets to local resources.

ABSTRACT WORD COUNT: 184

NOTE:

Figure number on first page: 3

LEGAL STATUS (Type, Pub Date, Kind, Text):

Assignee: 030423 A2 Transfer of rights to new applicant: Sun  
Microsystems, Inc. (2616592) 4150 Network  
Circle Santa Clara, California 95054 US

Application: 20000105 A2 Published application without search report

Examination: 040107 A2 Date of request for examination: 20031106

Search Report: 030521 A3 Separate publication of the search report

LANGUAGE (Publication,Procedural,Application): English; English; English

FULLTEXT AVAILABILITY:

Available Text	Language	Update	Word Count
CLAIMS A	(English)	200001	782
SPEC A	(English)	200001	13308



Total word count - document A            14090  
Total word count - document B            0  
Total word count - documents A + B    14090

...SPECIFICATION control is exercised over the way the multiple processors of a single processing set use **mutual exclusion** primitives ( **lmutexes** ). In practice it is the **various processing threads** in the processors which use the **mutexes** . In an MP machine, to provide a reasonably simple programming environment, the processors (or rather...

...Proper use of mutexes makes the processors of an MP system each act on its **own portion** of the total real state, with the important restriction that other processors will not modify...

...the real state, the only variable left undetermined is the order of acquisition of the **mutexes** by the various processors. If the processors (processor threads) in the **various processing** sets acquire and release **mutexes** in the same order, then all the modifications to the real state are wholly determined...voted and synchronized cycles on the I/O bus and so will automatically provide equivalent **mutex** ordering on **multiple processing** sets. No additional monitor capabilities are needed.

Yet another approach for the monitor unit to control **mutex** ordering is to use a combination of the above approaches. A relatively small number of...

17/5,K/3            (Item 3 from file: 348)  
DIALOG(R)File 348:EUROPEAN PATENTS  
(c) 2004 European Patent Office. All rts. reserv.

01106327

**Bus error handling in a computer system**

**Bus Fehlerbehandlung in einem Computersystem**

**Traitement d'erreur de bus dans un systeme d'ordinateur**

PATENT ASSIGNEE:

SUN MICROSYSTEMS, INC., (1392733), 901 San Antonio Road, Palo Alto,  
California 94303, (US), (Applicant designated States: all)

INVENTOR:

Williams, Emrys John, 1063 Morse Avenue No 3-205, Sunnyvale, California  
94089, (US)

LEGAL REPRESENTATIVE:

Harris, Ian Richard et al (72231), D. Young & Co., 21 New Fetter Lane,  
London EC4A 1DA, (GB)

PATENT (CC, No, Kind, Date): EP 969372 A2 000105 (Basic)

APPLICATION (CC, No, Date): EP 99304868 990622;

PRIORITY (CC, No, Date): US 106882 980630

DESIGNATED STATES: AT; BE; CH; CY; DE; DK; ES; FI; FR; GB; GR; IE; IT; LI;  
LU; MC; NL; PT; SE

EXTENDED DESIGNATED STATES: AL; LT; LV; MK; RO; SI

INTERNATIONAL PATENT CLASS: G06F-011/00; G06F-011/16

ABSTRACT EP 969372 A2

An I/O monitor includes an interface mechanism for connection between a processor and an I/O bus and an error signal modifier. The error signal modifier responds to an error signal from the I/O bus by substituting a determined response for passing to the processor. By returning a determined response to the processor, as opposed to the bus error signal, the need for bus error exception processing by the processor software is removed. The monitor determines a resource forming the source of the bus error and labels the resource as defective in a status register for the

resource in the monitor. The monitor generates an interrupt when a resource is first labelled as defective. Subsequently, further access to the resource by the processor are handled by the monitor. The monitor responds to an I/O read operation to a resource labelled as defective to prevent the I/O read operation from being passed to the bus and to return a determined data response. It responds to an I/O write operation to a resource labelled as defective to discard the I/O write operation and to terminate with an acknowledgement as the determined response.

ABSTRACT WORD COUNT: 189

NOTE:

Figure number on first page: 10

LEGAL STATUS (Type, Pub Date, Kind, Text):

Assignee: 030423 A2 Transfer of rights to new applicant: Sun Microsystems, Inc. (2616592) 4150 Network Circle Santa Clara, California 95054 US

Application: 20000105 A2 Published application without search report

LANGUAGE (Publication,Procedural,Application): English; English; English

FULLTEXT AVAILABILITY:

Available Text	Language	Update	Word Count
CLAIMS A	(English)	200001	1000
SPEC A	(English)	200001	13051
Total word count - document A			14051
Total word count - document B			0
Total word count - documents A + B			14051

...SPECIFICATION control is exercised over the way the multiple processors of a single processing set use **mutual exclusion** primitives ( **mutexes** ). In practice it is the **various processing threads** in the processors which use the **mutexes** . In an MP machine, to provide a reasonably simple programming environment, the processors (or rather...

...Proper use of mutexes makes the processors of an MP system each act on its **own portion** of the total real state, with the important restriction that other processors will not modify...

...the real state, the only variable left undetermined is the order of acquisition of the **mutexes** by the various processors. If the processors (processor threads) in the **various processing** sets acquire and release **mutexes** in the same order, then all the modifications to the real state are wholly determined...voted and synchronized cycles on the I/O bus and so will automatically provide equivalent **mutex** ordering on **multiple processing** sets. No additional monitor capabilities are needed.

Yet another approach for the monitor unit to control **mutex** ordering is to use a combination of the above approaches. A relatively small number of...

17/5,K/4 (Item 4 from file: 348)

DIALOG(R)File 348:EUROPEAN PATENTS

(c) 2004 European Patent Office. All rts. reserv.

01106326

**Control of multiple computer processes**

**Steuerung von mehreren Rechnerprozessen**

**Commande des processus d'ordinateur multiple**

PATENT ASSIGNEE:

Sun Microsystems, Inc., (2616592), 4150 Network Circle, Santa Clara, California 95054, (US), (Applicant designated States: all)

INVENTOR:

Williams, Emrys John, 1063 Morse Avenue no. 3-205, Sunnyvale, California  
94089, (US)

LEGAL REPRESENTATIVE:

Harris, Ian Richard et al (72231), D. Young & Co., 21 New Fetter Lane,  
London EC4A 1DA, (GB)

PATENT (CC, No, Kind, Date): EP 969369 A2 000105 (Basic)  
EP 969369 A3 040107

APPLICATION (CC, No, Date): EP 99304864 990622;

PRIORITY (CC, No, Date): US 107972 980630

DESIGNATED STATES: AT; BE; CH; CY; DE; DK; ES; FI; FR; GB; GR; IE; IT; LI;  
LU; MC; NL; PT; SE

EXTENDED DESIGNATED STATES: AL; LT; LV; MK; RO; SI

INTERNATIONAL PATENT CLASS: G06F-009/46

ABSTRACT EP 969369 A2

A program controlled apparatus includes one or more units for executing a **multiple process**. A **mutex** ordering mechanism controls the ordering of **mutex** ownership to provide deterministic execution of the processes. A **mutex** processor monitors **mutex** registers for determining **mutex** ownership. The **mutex** registers can be configured as sets of **mutex** request registers and **mutex** release registers. The apparatus may include a single processor configured to execute **multiple processes** concurrently, or **multiple processing** units, each configured to execute one or more processes. A monitor unit which can monitor equivalent operation of the processing sets can also include the **mutex** ordering mechanism.

ABSTRACT WORD COUNT: 102

NOTE:

Figure number on first page: 12

LEGAL STATUS (Type, Pub Date, Kind, Text):

Assignee: 030423 A2 Transfer of rights to new applicant: Sun  
Microsystems, Inc. (2616592) 4150 Network  
Circle Santa Clara, California 95054 US

Application: 20000105 A2 Published application without search report

Search Report: 040107 A3 Separate publication of the search report

LANGUAGE (Publication,Procedural,Application): English; English; English

FULLTEXT AVAILABILITY:

Available Text	Language	Update	Word Count
CLAIMS A	(English)	200001	671
SPEC A	(English)	200001	13522
Total word count - document A			14193
Total word count - document B			0
Total word count - documents A + B			14193

...ABSTRACT A2

A program controlled apparatus includes one or more units for executing a **multiple process**. A **mutex** ordering mechanism controls the ordering of **mutex** ownership to provide deterministic execution of the processes. A **mutex** processor monitors **mutex** registers for determining **mutex** ownership. The **mutex** registers can be configured as sets of **mutex** request registers and **mutex** release registers. The apparatus may include a single processor configured to execute **multiple processes** concurrently, or **multiple processing** units, each configured to execute one or more processes. A monitor unit which can monitor equivalent operation of the processing sets can also include the **mutex** ordering mechanism.

...SPECIFICATION there is provided a program controlled apparatus comprising at least one execution unit for executing **multiple**

programmed **processes** and a **mutual exclusion** primitive ( **mutex** ) ordering mechanism controlling the ordering of **mutex** ownership to provide deterministic execution of the processes.

By controlling the order of **mutex** ownership, it is possible to control the execution of the processes to achieve deterministic execution therefor. This can enable fault tolerance to be built into **many multi - process ( multi - threaded ) processing** environments including, for example, networked fault tolerant systems.

A **mutex** processor can be operable to monitor **mutex** registers for determining **mutex** ownership. By controlling the access to the **mutex** registers, that is the ownership thereof, a...

...of **mutex** processing can be achieved.

The **mutex** registers can be configured as sets of **mutex** request registers and **mutex** release registers.

The invention finds application, for example, to a single processor configured to process **multiple threads** , or processes concurrently. The processes could, for example, be operating system processes. The invention also...

...be provided for monitoring equivalent operation of the processing sets, the monitor unit comprising the **mutex** ordering mechanism.

In accordance with another aspect of the invention, there is provided computing apparatus including a **plurality of processing** sets, wherein at least a first processing set is operable asynchronously of a second processing...

...provided with multiple sets of **mutex** start registers and a hash mechanism for accessing a **mutex** list for an I/O cycle.

In accordance with another aspect of the invention, there is provided a method of providing deterministic execution of **multiple processes** , the method comprising:

executing the processes; and

controlling the ordering of **mutexes** to provide deterministic execution of the processes.

In accordance with yet a further aspect of...control is exercised over the way the multiple processors of a single processing set use **mutual exclusion** primitives ( **mutexes** ). In practice it is the **various processing threads** in the processors which use the **mutexes** . In an MP machine, to provide a reasonably simple programming environment, the processors (or rather...

...Proper use of **mutexes** makes the processors of an MP system each act on its **own portion** of the total real state, with the important restriction that other processors will not modify...

...the real state, the only variable left undetermined is the order of acquisition of the **mutexes** by the various processors. If the processors (processor threads) in the **various processing** sets acquire and release **mutexes** in the same order, then all the modifications to the real state are wholly determined...voted and synchronized cycles on the I/O bus and so will automatically provide equivalent **mutex** ordering on **multiple processing** sets. No additional monitor capabilities are needed.

Yet another approach for the monitor unit to control **mutex** ordering is to use a combination of the above approaches. A relatively small number of...for bus operations.

There has been described a program controlled apparatus including means for executing **multiple processes** and means for controlling an ordering of **mutex** ownership for the processes to provide deterministic execution of the processes.

Although an embodiment of...in the form of separate electronics or



software which is operable to capture requests for **mutex** allocation and to control the ordering of the allocation of the **mutexes** for the **various threads** .

Accordingly, it is to be understood that the invention is applicable to any **multi - threaded , multi - process** or multi-processor system which employs **mutexes** for controlling access to common system resources, such as, for example, memory. The invention finds application, for example, to a single processor configured to process **multiple processes** , or threads, concurrently. The invention also finds application to a plurality of processing units, each...

- CLAIMS 1. A program controlled apparatus including means for executing **multiple processes** and means for controlling an ordering of **mutex** ownership for the processes to provide deterministic execution of the processes.
2. The apparatus of Claim 1, comprising:
    - at least one unit for executing **multiple** programmed **processes** ; and
    - a **mutex** ordering mechanism controlling an ordering of **mutex** ownership for the processes to provide deterministic execution of the **multiple processes** .
  3. The apparatus of claim 1 or claim 2, wherein the **mutex** ordering mechanism comprises at least one **mutex** register and a **mutex** processor operable to monitor the at least one mutex register for determining mutex ownership.
  4. The apparatus of claim 3, wherein the **mutex** ordering mechanism comprises sets of **mutex** request registers and **mutex** release registers.
  5. The apparatus of any preceding claim, comprising a processor configured to process **multiple processes** concurrently.
  6. The apparatus of any one of claims 1 to 4, comprising a plurality...
- ...monitor unit for monitoring equivalent operation of the processing sets, the monitor unit comprising said **mutex** ordering mechanism.
9. Computing apparatus, comprising:
    - a **plurality** of **processing** sets, wherein at least a first processing set is operable asynchronously of a second processing...

?

File 347:JAPIO Oct 1976-2003/Oct(Updated 040202)

(c) 2004 JPO & JAPIO

File 350:Derwent WPIX 1963-2004/UD,UM &UP=200409

(c) 2004 Thomson Derwent

Set	Items	Description
S1	30	MUTEX? OR MUTUAL()EXCLUSION? OR MUT()EX OR MUTUALEXCLUSION?
S2	3684828	THREAD???? ? OR PROCESS OR PROCESSES OR PROCESSING OR TASK- ???? ?
S3	20426	(PLURALITY OR MANY OR MULTI OR SEVERAL OR NUMEROUS OR MULT- IPLE OR MULTITUD? OR MULTIPLICITY OR PLURIF? OR VARIOUS OR VA- RIETY) (1W)S2
S4	31328	(GROUP???? ? OR CLUSTER??? ? OR NUMBER OR NETWORK? ? OR CH- AIN? ? OR SERIES OR THREE OR COLLECTION? OR THIRD OR ASSORT? - OR RANGE? ?) (1W)S2
S5	2461	MULTITHREAD? OR MULTIPROCESS OR MULTIPROCESSES OR MULTIPRO- CESSING OR MULTITASK?
S6	0	GHOSTLOCK? OR GHOST()LOCK??? ?
S7	98136	OWN OR OWNS OR OWNER? OR OWNED
S8	435	OWNING
S9	114	CO()S7:S8 OR S7:S8(2N) (TEMPORAR??? ? OR BRIEF?? ? OR INTER- IM OR MOMENTAR? OR TRANSIT?)
S10	996	S7:S8(2N) (PARTIAL? OR PORTION? OR PART OR DIVID?? ? OR DIV- ISION? OR SUBDIVID? OR SUBDIVISION? OR PARTITION? OR REDIVID? OR REDIVISION? OR SPLIT???? ?)
S11	0	PARTOWNER? OR COOWN?
S12	59	(PLURALITY OR MANY OR MULTI OR SEVERAL OR NUMEROUS OR MULT- IPLE OR MULTITUD? OR MULTIPLICITY OR PLURIF? OR VARIOUS OR VA- RIETY) (1W)S7:S8
S13	242	(GROUP???? ? OR CLUSTER??? ? OR NUMBER OR NETWORK? ? OR CH- AIN? ? OR SERIES OR THREE OR COLLECTION? OR THIRD OR ASSORT? - OR RANGE? ?) (1W)S7:S8
S14	89	MUTUAL(1W)EXCLUSION?
S15	26	(S1 OR S14) AND S3:S5
S16	0	S15 AND S9:S13
S17	26	IDPAT S15 (sorted in duplicate/non-duplicate order)
S18	24	IDPAT S15 (primary/non-duplicate records only)
?		

? t18/9/all

18/9/1 (Item 1 from file: 350)  
DIALOG(R)File 350:Derwent WPIX  
(c) 2004 Thomson Derwent. All rts. reserv.

015900698 \*\*Image available\*\*  
WPI Acc No: 2004-058537/200406  
XRPX Acc No: N04-047300

**Request control method for computer system, involves assuring that thread associated with target mutex , is not operating, when target mutex is in specific mode**

Patent Assignee: HEWLETT-PACKARD CO LP (HEWP )  
Inventor: BURROWS M; GHEMAWAT S; VANDERVOORDE M T  
Number of Countries: 001 Number of Patents: 001  
Patent Family:  
Patent No Kind Date Applicat No Kind Date Week  
US 6662364 B1 20031209 US 99434831 A 19991105 200406 B

Priority Applications (No Type Date): US 99434831 A 19991105

Patent Details:

Patent No Kind Lan Pg Main IPC Filing Notes  
US 6662364 B1 18 G06F-009/00

Abstract (Basic): US 6662364 B1

NOVELTY - A target **mutex** which is capable of specifying different synchronization modes, is determined. When the target **mutex** is in specific mode, an atomic machine instruction assures that thread associated with the target **mutex** , is not operating, based on which a request is granted, to acquire the target **mutex** .

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

- (1) computer system; and
- (2) computer program product for controlling request

USE - For controlling request to acquire target **mutex** in computer system (claimed).

ADVANTAGE - Enables reducing the cost of synchronization in **multi threaded** environment, regardless of number of processors.

DESCRIPTION OF DRAWING(S) - The figure shows a flowchart explaining the request control method.

pp; 18 DwgNo 2/6

Title Terms: REQUEST; CONTROL; METHOD; COMPUTER; SYSTEM; ASSURE; THREAD;

ASSOCIATE; TARGET; OPERATE; TARGET; SPECIFIC; MODE

Derwent Class: T01

International Patent Class (Main): G06F-009/00

File Segment: EPI

Manual Codes (EPI/S-X): T01-F02C2; T01-S03

18/9/2 (Item 2 from file: 350)  
DIALOG(R)File 350:Derwent WPIX  
(c) 2004 Thomson Derwent. All rts. reserv.

015875860 \*\*Image available\*\*  
WPI Acc No: 2004-033691/200403  
XRPX Acc No: N04-026772

**Robot control system for manufacturing environment, has mutual exclusion mechanism to prevent simultaneous access by more than one robot control program to given robotic machine**

Patent Assignee: KNEIFEL R W (KNEI-I); STODDARD K A (STOD-I)

Inventor: KNEIFEL R W; STODDARD K A

Number of Countries: 001 Number of Patents: 001

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 20030220715	A1	20031127	US 2002153537	A	20020522	200403 B

Priority Applications (No Type Date): US 2002153537 A 20020522

Patent Details:

Patent No	Kind	Lan Pg	Main IPC	Filing Notes
US 20030220715	A1		9 G06F-019/00	

Abstract (Basic): US 20030220715 A1

NOVELTY - Robot controllers (34) have an associated motion system (36) adapted to control robotic machine (38). A remote instruction source (32) supplies motion control commands over computer network (40) to selected controllers. The **mutual exclusion** mechanism residing on either the instruction source or one of the controllers, prevents simultaneous access by more than one control program to a given robotic machine.

USE - For controlling multiple robotic machines used for performing **various** manufacturing **tasks** in workstation.

ADVANTAGE - The **mutual exclusion** mechanism enables exchanging control of a given robot dynamically between various control programs running on different robot.

DESCRIPTION OF DRAWING(S) - The figure shows the block diagram of the robot control system.

remote instruction source (32)

robot controller (34)

associated motion system (36)

robotic machine (38)

computer network (40)

pp; 9 DwgNo 3/5

Title Terms: ROBOT; CONTROL; SYSTEM; MANUFACTURE; ENVIRONMENT; MUTUAL; EXCLUDE; MECHANISM; PREVENT; SIMULTANEOUS; ACCESS; MORE; ONE; ROBOT; CONTROL; PROGRAM; ROBOT; MACHINE

Derwent Class: T01; T06

International Patent Class (Main): G06F-019/00

File Segment: EPI

Manual Codes (EPI/S-X): T01-J07B; T06-D07B

**18/9/3 (Item 3 from file: 350)**

DIALOG(R)File 350:Derwent WPIX

(c) 2004 Thomson Derwent. All rts. reserv.

015741975 \*\*Image available\*\*

WPI Acc No: 2003-804176/200375

Related WPI Acc No: 2003-788379

XRPX Acc No: N03-644636

Multithreaded **application program replicating method**, involves **piggybacking** mutex ordering information onto multicast messages **specifying order in which threads in primary replica are granted their claims to mutexes**

Patent Assignee: ETERNAL SYSTEMS INC (ETER-N)

Inventor: MELLIAR-SMITH P M; MOSER L E

Number of Countries: 103 Number of Patents: 001

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
WO 200384116	A1	20031009	WO 2003US9443	A	20030325	200375 B

Priority Applications (No Type Date): US 2002367616 P 20020325; US



2002367615 P 20020325

Patent Details:

Patent No Kind Lan Pg Main IPC Filing Notes

WO 200384116 A1 E 82 H04L-001/22

Designated States (National): AE AG AL AM AT AU AZ BA BB BG BR BY BZ CA  
CH CN CO CR CU CZ DE DK DM DZ EC EE ES FI GB GD GE GH GM HR HU ID IL IN  
IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX MZ NI NO  
NZ OM PH PL PT RO RU SC SD SE SG SK SL TJ TM TN TR TT TZ UA UG US UZ VC  
VN YU ZA ZM ZW

Designated States (Regional): AT BE BG CH CY CZ DE DK EA EE ES FI FR GB  
GH GM GR HU IE IT KE LS LU MC MW MZ NL OA PT RO SD SE SI SK SL SZ TR TZ  
UG ZM ZW

Abstract (Basic): WO 200384116 A1

NOVELTY - The method involves piggybacking **mutex** ordering information onto regular multicast messages specifying an order in which threads in a primary replica have been granted their claims to **mutexes**. The messages containing the **mutex** ordering information are received to determine the order in which threads in a backup replica are granted the **mutexes**.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

(a) a system for replicating a **multithreaded** application program using a semi-active or a passive replication strategy

(b) a software mechanism for replicating a **multithreaded** application program using a semi-active or passive replication strategy

USE - Used for replicating **multithreaded** application programs in computer system.

ADVANTAGE - The method exploits a reliable source-ordered multicast protocol to maintain strong replica consistency without the need to count instruction, add significant messaging overhead and modify application code.

DESCRIPTION OF DRAWING(S) - The drawing shows a primary replica and a backup replica.

Replicas (10)

Primary replica (12)

Backup replica (14)

Threads (16)

Shared resource (40)

pp; 82 DwgNo 1/10

Title Terms: APPLY; PROGRAM; REPLICA; METHOD; ORDER; INFORMATION; MESSAGE; SPECIFIED; ORDER; THREAD; PRIMARY; REPLICA; CLAIM

Derwent Class: T01

International Patent Class (Main): H04L-001/22

File Segment: EPI

Manual Codes (EPI/S-X): T01-F02B; T01-S03

18/9/4 (Item 4 from file: 350)

DIALOG(R)File 350:Derwent WPIX

(c) 2004 Thomson Derwent. All rts. reserv.

015726179 \*\*Image available\*\*

WPI Acc No: 2003-788379/200374

Related WPI Acc No: 2003-804176

XRPX Acc No: N03-631686

Multithreaded application program replicating method, involves delivering control messages using multicast group communication protocol that delivers messages in order

Patent Assignee: ETERNAL SYSTEMS INC (ETER-N)

Inventor: MELLIAR-SMITH P M; MOSER L E

Number of Countries: 103 Number of Patents: 001

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
WO 200383614	A2	20031009	WO 2003US9233	A	20030324	200374 B

Priority Applications (No Type Date): US 2002367616 P 20020325; US

2002367615 P 20020325

Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
-----------	------	-----	----	----------	--------------

WO 200383614	A2	E	63	G06F-000/00	
--------------	----	---	----	-------------	--

Designated States (National): AE AG AL AM AT AU AZ BA BB BG BR BY BZ CA CH CN CO CR CU CZ DE DK DM DZ EC EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX MZ NI NO NZ OM PH PL PT RO RU SC SD SE SG SK SL TJ TM TN TR TT TZ UA UG US UZ VC VN YU ZA ZM ZW

Designated States (Regional): AT BE BG CH CY CZ DE DK EA EE ES FI FR GB GH GM GR HU IE IT KE LS LU MC MW MZ NL OA PT RO SD SE SI SK SL SZ TR TZ UG ZM ZW

Abstract (Basic): WO 200383614 A2

NOVELTY - The method involves multicasting control messages at each replica. The control messages are delivered using a multicast group communication protocol that delivers the messages in an order that determines the order in which an operating systems thread library grants claims of **mutexes** to threads in the replicas.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

(a) a method of achieving strong replica consistency for replicated **multithreaded** application programs

(b) a software mechanism for replicating a **multithreaded** application program

(c) a system for executing threads in replicas of an application program within a computing environment

(d) a computer readable media and a computer for **multithreaded** execution and communication with multiple program replicas.

USE - Used for **multithreaded** application programs that replicate using egalitarian and competitive active replication strategy.

ADVANTAGE - The method achieves strong replica consistency of **multithreaded** application programs that are replicated using the egalitarian and competitive active replication strategy, thereby allowing maximum degree of concurrency.

DESCRIPTION OF DRAWING(S) - The drawing shows two replicas each executing **multiple threads** that share data, which are protected by **mutexes**.

Threads (20-26)

Multicast group communication protocol (36)

Shared resource (40)

Shared data (44)

**Mutex** (46)

pp; 63 DwgNo 1/6

Title Terms: APPLY; PROGRAM; REPLICA; METHOD; DELIVER; CONTROL; MESSAGE; GROUP; COMMUNICATE; PROTOCOL; DELIVER; MESSAGE; ORDER

Derwent Class: T01

International Patent Class (Main): G06F-000/00

File Segment: EPI

Manual Codes (EPI/S-X): T01-F02C2; T01-F03B; T01-S03

DIALOG(R)File 350:Derwent WPIX  
(c) 2004 Thomson Derwent. All rts. reserv.

015669696 \*\*Image available\*\*

WPI Acc No: 2003-731883/200369

Related WPI Acc No: 2003-577937; 2003-577938; 2003-577939; 2003-587336;  
2003-587337; 2003-587342; 2003-598605; 2003-646438; 2003-646506

XRPX Acc No: N03-584974

**Shared resources e.g. counter, access synchronization method, involves  
allowing thread to access shared resource according to synchronization  
primitive e.g. mutex , of access manager**

Patent Assignee: IDETIC INC (IDET-N)

Inventor: DE BONET J S

Number of Countries: 102 Number of Patents: 001

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
WO 200377127	A2	20030918	WO 2003US1153	A	20030115	200369 B

Priority Applications (No Type Date): US 2003345067 A 20030115; US  
2002348566 P 20020115; US 2002348707 P 20020115; US 2002349344 P 20020118  
; US 2002349424 P 20020118

Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
-----------	------	-----	----	----------	--------------

WO 200377127	A2	E	56	G06F-009/46	
--------------	----	---	----	-------------	--

Designated States (National): AE AG AL AM AT AU AZ BA BB BG BR BY BZ CA  
CH CN CO CR CU CZ DE DK DM DZ EC EE ES FI GB GD GE GH GM HR HU ID IL IN  
IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX MZ NO NZ  
OM PH PL PT RO RU SC SD SE SG SK SL TJ TM TN TR TT TZ UA UG US UZ VC VN  
YU ZA ZM ZW

Designated States (Regional): AT BE BG CH CY CZ DE DK EA EE ES FI FR GB  
GH GM GR HU IE IT KE LS LU MC MW MZ NL OA PT SD SE SI SK SL SZ TR TZ UG  
ZM ZW

Abstract (Basic): WO 200377127 A2

NOVELTY - The method involves receiving a request for a shared  
resource e.g. counter, that is associated with an access manager having  
a thread synchronization primitive e.g. **mutex** , from a thread. The  
thread is then allowed to access the resource according to the  
synchronization primitive.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the  
following:

(a) a data processing system readable medium with codes for  
performing shared resource access synchronizing method

(b) a system for the protection of the shared resource

(c) a system for managing shared resources

(d) a method for managing shared resources.

USE - Used for controlling access to shared resources e.g. counter,  
across **multiple threads** .

ADVANTAGE - The accessing of shared resources based on thread  
synchronization primitive prevents them from inadvertent corruption in  
a **multithread** environment.

DESCRIPTION OF DRAWING(S) - The drawing shows a computer that can  
be used in a shared resource access synchronizing method. S  
pp; 56 DwgNo 1/9

Title Terms: SHARE; RESOURCE; COUNTER; ACCESS; SYNCHRONISATION; METHOD;  
ALLOW; THREAD; ACCESS; SHARE; RESOURCE; ACCORD; SYNCHRONISATION;  
PRIMITIVE; ACCESS; MANAGE

Derwent Class: T01

International Patent Class (Main): G06F-009/46

File Segment: EPI

Manual Codes (EPI/S-X): T01-F02C; T01-F02C1; T01-F03B; T01-H03D

18/9/6 (Item 6 from file: 350)  
DIALOG(R)File 350:Derwent WPIX  
(c) 2004 Thomson Derwent. All rts. reserv.

014981533 \*\*Image available\*\*  
WPI Acc No: 2003-042048/200304  
XRAM Acc No: C03-010313  
XRPX Acc No: N03-032973

**Management of Laboratory Automation Workcell System comprises associating a sample protocol to each sample, associating a resource driver to each resource and executing sample protocols with a process controller**

Patent Assignee: INPECO SRL (INPE-N); FAVA D (FAVA-I); GARGHENTINO E (GARG-I); PEDRAZZINI G (PEDR-I)

Inventor: FAVA D; GARGHENTINO E; PEDRAZZANI G; PEDRAZZINI G

Number of Countries: 027 Number of Patents: 002

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
EP 1248170	A1	20021009	EP 2001830237	A	20010405	200304 B
US 20020147515	A1	20021010	US 2002113575	A	20020402	200304

Priority Applications (No Type Date): EP 2001830237 A 20010405

Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
-----------	------	-----	----	----------	--------------

EP 1248170	A1	E	19	G05B-019/418	
------------	----	---	----	--------------	--

Designated States (Regional): AL AT BE CH CY DE DK ES FI FR GB GR IE IT  
LI LT LU LV MC MK NL PT RO SE SI TR

US 20020147515	A1	G06F-019/00
----------------	----	-------------

Abstract (Basic): EP 1248170 A1

NOVELTY - Management of Workcell Systems using an Automation Management System comprises associating a Sample Protocol to each Sample, associating a Resource Driver controlled by a Process Controller and communicating with other Resource Drivers to each Resource and executing the Sample Protocols by the Process Controller.

DETAILED DESCRIPTION - Management of Workcell Systems using an Automation Management System (AMS) to control Resources (R1-Rn) designated to handle Samples (S1) along the Workcell Systems to perform operations on the Samples comprises:

- (a) associating a Sample Protocol (SP1) to each Sample;
- (b) associating a Resource Driver (D1-Dn) to each Resource; and
- (c) executing Sample Protocols by a Process Controller (PC).

Each Resource Driver is controlled by the Process Controller. It communicates with other Resource Drivers through variables associated to each Resource. The variables define and modify the status of the Resources.

An INDEPENDENT CLAIM is included for an Automation Management System comprising mechanism to transfer elements between equipment where each element has a protocol and an execution status associated to it.

USE - For managing a Laboratory Automation Workcell System where each Sample is a biological specimen collected in a Sample Container. It is for controlling an In Vitro Diagnostics (IVD) Workcell that performs sample treatment and test analyses on biological samples using Diagnostic Instruments as Sample Assay Devices, a High Throughput Screening (HTS) Laboratory Automation Workcell System that performs a big amount of screening analyses of biological samples on Sample Assay Devices following different protocols, in order to generate data used for pharmaceutical research, a Nucleic Acid Technology (NAT) Workcell that performs sample treatment and test analyses on nucleic acids of

biological samples using appropriate Instruments as Sample Assay Devices, a Bio-Repository Laboratory Automation Workcell System used as a permanent storage for biological samples that may be kept refrigerated for years and used upon request; and an Industrial Control System. (All claimed).

ADVANTAGE - The process provides a Laboratory Automation Workcell System for medium/high throughput situations where very high flexibility and adaptability are required. The reactivity of the systems makes it possible to support many different assay formats sequentially or in parallel and immediate adaptability to program changes and unforeseen circumstances around and during a program run.

DESCRIPTION OF DRAWING(S) - The figure shows a general structure of a Workcell System of the invention.

- Automation Management System (AMS)
- Resource Driver (D1-Dn)
- Data Manager (DM)
- Graphic User Interface (GUI)
- Laboratory Information System Driver (LD)
- Laboratory Information System (LIS)
- Process Controller (PC)
- Resources (R1-Rn)
- Sample (S1)
- Sample Protocol (SP1)
- User (U)

pp; 19 DwgNo 1/4

#### Technology Focus:

TECHNOLOGY FOCUS - COMPUTING AND CONTROL - Preferred Process: The activities of the Workcell System are coordinated by a Process Controller capable of importing Sample Protocols and executing these protocols in a **multi - tasking** way, automatically taking the appropriate decisions according to the variables of the Resources.

Preferred Components: A Data Manager (DM) as the handler of Sample data and Sample Protocols operates in coordination with the other Resources by a set of variables used for communication. A Database associated to the Data Manager represents a collection of Sample Protocols and Sample Results obtained from Assay devices or calculated by Data Manager using defined mathematical formulae applied on other data included in the results. A Graphic User Interface (GUI) as the handler of communication with the User (U) operates in coordination with other Resources by a set of variables indicating the status of the Resource itself.

A Laboratory Information System Driver (LD) as the handler of communication with the Laboratory Information System (LIS) operates in coordination with the other Resources by a set of variables indicating the status of the Driver itself. The Resources include Sample Carrier Devices, Sample Treatment Devices, Sample Container Handling Devices, and Sample Assay Devices . Each Sample Protocol is associated to a Priority Level updatable during the execution of the Protocol. Each Resource used in a Sample Protocol is associated to a Wait Parameter to indicate the time when the Resource completes its task in that Sample Protocol and a Critical Time Parameter at the end of which the Resource must have absolute priority with respect to any other Protocol. Each Sample Protocol is associated to a Bottleneck Parameter that can be set up in a way to provide a negative and a positive feedback loop that can optimize the throughput on the system. The communication among the Resources and the Process Controller is driven by a set of variables, representing the Resource Logical Status (RLS) that expresses the various logical conditions of the associated Resource and the Resource Physical Status (RPS) expressing the various physical conditions of the associated Resource.

The Automation Management System controls and integrates the

Resources in order to have a homogeneous and concurrent treatment of Samples along the Workcell System by associating to each Sample Container a Sample Protocol and executing all the Protocols in a concurrent way guaranteeing **multi - threading** , bottleneck avoidance, dynamic assignment of Resources in **mutual exclusion** , sample priority handling, Resource load balancing and automatic error recovery.

Title Terms: MANAGEMENT; LABORATORY; AUTOMATIC; SYSTEM; COMPRISE; ASSOCIATE ; SAMPLE; PROTOCOL; SAMPLE; ASSOCIATE; RESOURCE; DRIVE; RESOURCE; EXECUTE ; SAMPLE; PROCESS; CONTROL

Derwent Class: B04; D16; J04; S03; T06

International Patent Class (Main): G05B-019/418; G06F-019/00

International Patent Class (Additional): G01N-035/00; G05B-019/042

File Segment: CPI; EPI

Manual Codes (CPI/A-N): B11-C09; B12-K04E; D05-H09; J04-B01

Manual Codes (EPI/S-X): S03-E14H; S03-E15; T06-A04A2A; T06-A04B1; T06-A04B7

Chemical Fragment Codes (M6):

\*01\* M905 P831 Q233 R501 R511 R515 R521 R528

18/9/7 (Item 7 from file: 350)

DIALOG(R)File 350:Derwent WPIX

(c) 2004 Thomson Derwent. All rts. reserv.

014868927 \*\*Image available\*\*

WPI Acc No: 2002-689633/200274

XRPX Acc No: N02-543910

**Memory block allocation method in multi - task operating system, involves allocating small portion of cached memory block in cache slot to thread, when block size is not more than predetermined threshold**

Patent Assignee: HEWLETT-PACKARD CO (HEWP )

Inventor: KARKARE A; MCGOWEN A

Number of Countries: 001 Number of Patents: 001

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 6427195	B1	20020730	US 2000592781	A	20000613	200274 B

Priority Applications (No Type Date): US 2000592781 A 20000613

Patent Details:

Patent No	Kind	Lan Pg	Main IPC	Filing Notes
US 6427195	B1	14	G06F-012/06	

Abstract (Basic): US 6427195 B1

NOVELTY - A cache slot having cached memory block previously freed by a thread, is provided privately to the thread. In response to the memory allocation request from the thread, a small portion of the cached memory block of cache slot is allocated, if the size of memory block is not more than a predetermined threshold.

DETAILED DESCRIPTION - An INDEPENDENT CLAIM is included for computer readable memory medium storing memory block allocation program.

USE - In **multi - tasking** operating systems.

ADVANTAGE - Provides efficient system for allocating memory blocks to **multiple threads** . Eliminates the need for coalescence, hence optimal **multi - threading** is enabled and avoids **MUTEX** locking.

DESCRIPTION OF DRAWING(S) - The figure shows the flow diagram of ordinary block cache allocation process.

pp; 14 DwgNo 3B/5

Title Terms: MEMORY; BLOCK; ALLOCATE; METHOD; MULTI; TASK; OPERATE; SYSTEM; ALLOCATE; PORTION; MEMORY; BLOCK; CACHE; SLOT; THREAD; BLOCK; SIZE; MORE;

PREDETERMINED; THRESHOLD  
Derwent Class: T01  
International Patent Class (Main): G06F-012/06  
File Segment: EPI  
Manual Codes (EPI/S-X): T01-F02C2; T01-F05E; T01-H03A; T01-S03

18/9/8 (Item 8 from file: 350)  
DIALOG(R)File 350:Derwent WPIX  
(c) 2004 Thomson Derwent. All rts. reserv.

014625367 \*\*Image available\*\*  
WPI Acc No: 2002-446071/200248  
XRPX Acc No: N02-351453

**Resource access control mechanism for multi - thread computing environment, suspends new thread until previous mutex is unlocked in response to previous thread finishing access to resource**

Patent Assignee: SUN MICROSYSTEMS INC (SUNM ); LATOUR J (LATO-I)

Inventor: LATOUR J

Number of Countries: 026 Number of Patents: 002

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
EP 1199632	A1	20020424	EP 2000309238	A	20001020	200248 B
US 20020078123	A1	20020620	US 2001928618	A	20010813	200248

Priority Applications (No Type Date): EP 2000309238 A 20001020

Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
-----------	------	-----	----	----------	--------------

EP 1199632	A1	E	20	G06F-009/46	
------------	----	---	----	-------------	--

Designated States (Regional): AL AT BE CH CY DE DK ES FI FR GB GR IE IT

LI LT LU LV MC MK NL PT RO SE SI

US 20020078123	A1			G06F-009/00	
----------------	----	--	--	-------------	--

Abstract (Basic): EP 1199632 A1

NOVELTY - A new **mutex** for thread attempting an access to a resource, is locked. If previous **mutex** is already locked, the new thread is suspended until the previous **mutex** is unlocked in response to a previous thread finishing access to the resource.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are included for the following:

- (1) Resource access control program for a **multi - thread** computing environment;
- (2) Computer program product for resource access control;
- (3) Computer system;
- (4) Resource access control method for **multi - thread** computing environment.

USE - For **multi - threaded** programming environment.

ADVANTAGE - By ensuring that only one thread is waiting on the release of any one **mutex** , problems with ordering of the resource access is avoided.

DESCRIPTION OF DRAWING(S) - The figure shows the flow diagram illustrating the control of a sequence of **mutexes** .

pp; 20 DwgNo 10/12

Title Terms: RESOURCE; ACCESS; CONTROL; MECHANISM; MULTI; THREAD;  
COMPUTATION; ENVIRONMENT; SUSPENSION; NEW; THREAD; UNLOCK; RESPOND;  
THREAD; FINISH; ACCESS; RESOURCE

Derwent Class: T01

International Patent Class (Main): G06F-009/00; G06F-009/46

File Segment: EPI

Manual Codes (EPI/S-X): T01-F02C2; T01-S03

18/9/9 (Item 9 from file: 350)  
DIALOG(R)File 350:Derwent WPIX  
(c) 2004 Thomson Derwent. All rts. reserv.

013990619 \*\*Image available\*\*  
WPI Acc No: 2001-474834/200151  
XRPX Acc No: N01-351428

**Computer system for multithreading environments, has thread library  
differentiating between high priority process and lightweight process for  
allowing lightweight process to complete started process**

Patent Assignee: SUN MICROSYSTEMS INC (SUNM )  
Inventor: TUCKER A G  
Number of Countries: 001 Number of Patents: 001  
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 6223204	B1	20010424	US 96767353	A	19961218	200151 B
			US 96767353	A	19961218	

Priority Applications (No Type Date): US 96767353 A 19961218; US 96767353 A  
19961218

Patent Details:

Patent No	Kind	Lan Pg	Main IPC	Filing Notes
US 6223204	B1	11	G06F-009/00	Cont of application US 96767353

Abstract (Basic): US 6223204 B1

NOVELTY - User level memory (22a) has **multiple threads** (25a-25f) and kernel level memory (22b) has lightweight processes (LWP) (32a-32d). The thread enters into spin state when scheduled onto a LWP that is running while it enters into sleep state when LWP is in non-running state. Thread library distinguishes between high priority and LWP that are selected and places high priority process into sleep state whereas LWP is allowed to complete the process started.

DETAILED DESCRIPTION - An INDEPENDENT CLAIM is also included for the scheduling method of thread onto a LWP in computer system.

USE - For allocating resources in **multithreading** computing environment.

ADVANTAGE - Allocation of resource is effective by having thread library which indicates the states of threads.

DESCRIPTION OF DRAWING(S) - The figure shows the block diagram of computer system implementing **mutex** adaptive locking.

User level memory (22a)  
Kernel level memory (22b)  
Threads (25a-25f)  
LWP (32a-32d)  
pp; 11 DwgNo 2/4

Title Terms: COMPUTER; SYSTEM; ENVIRONMENT; THREAD; LIBRARY; DIFFERENTIAL;  
HIGH; PRIORITY; PROCESS; LIGHT; PROCESS; ALLOW; LIGHT; PROCESS; COMPLETE;  
START; PROCESS

Derwent Class: T01

International Patent Class (Main): G06F-009/00

File Segment: EPI

Manual Codes (EPI/S-X): T01-F02A; T01-F02C2; T01-F05A; T01-F07; T01-J20B2

18/9/10 (Item 10 from file: 350)  
DIALOG(R)File 350:Derwent WPIX  
(c) 2004 Thomson Derwent. All rts. reserv.

013850259 \*\*Image available\*\*



WPI Acc No: 2001-334472/200135

XRPX Acc No: N01-241330

**Program operating method, involves linking different libraries to program via common API to allow the same thread function calls to operate in either multithreaded or non-threaded modes**

Patent Assignee: HEWLETT-PACKARD CO (HEWP )

Inventor: KIRSHENBAUM E; MOORE K; UNDERWOOD W R

Number of Countries: 001 Number of Patents: 001

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 6173309	B1	20010109	US 97803404	A	19970220	200135 B

Priority Applications (No Type Date): US 97803404 A 19970220

Patent Details:

Patent No	Kind	Lan Pg	Main IPC	Filing Notes
US 6173309	B1	11	G06F-009/00	

Abstract (Basic): US 6173309 B1

NOVELTY - The different libraries are linked to the program via a common application programming interface allowing the same thread function calls to operate in either **multithreaded** or non-threaded modes. The null thread library implements thread function calls as blocking functions that synchronize thread launches to eliminate the required performance and memory overhead during **multi - threaded processes**.

DETAILED DESCRIPTION - A program is operated in either a **multithreaded** or non-threaded programming environment by linking the program to different libraries. A standard thread library is linked to the program when operated in a **multithreaded** mode. A null thread library is then linked to the program to operate in a non-threaded mode.

INDEPENDENT CLAIM is also included for a program operating system.

USE - Program operating method.

ADVANTAGE - Ensures **mutual exclusion** and maintains thread-specific data by abstracting thread model into a single interface providing objects and facilities for launching threads. The single interface abstracts away the peculiarities of any particular thread package. Enables programmer to create a single program written in terms of the interface which is able to work with any of the thread packages.

DESCRIPTION OF DRAWING(S) - The figure shows the operational diagram of the program operating system.

pp; 11 DwgNo 3/6

Title Terms: PROGRAM; OPERATE; METHOD; LINK; PROGRAM; COMMON; ALLOW; THREAD ; FUNCTION; CALL; OPERATE; NON; THREAD; MODE

Derwent Class: T01

International Patent Class (Main): G06F-009/00

File Segment: EPI

Manual Codes (EPI/S-X): T01-J20B1; T01-J20B2; T01-S01C

**18/9/11 (Item 11 from file: 350)**

DIALOG(R)File 350:Derwent WPIX

(c) 2004 Thomson Derwent. All rts. reserv.

013514549 \*\*Image available\*\*

WPI Acc No: 2000-686495/200067

XRPX Acc No: N00-507546

**Life cycle controlling method for object in multithread computing environment, involves setting count associated with object**

**identification, in table to one and unlocking mutex lock**

Patent Assignee: IONA TECHNOLOGIES INC (IONA-N)

Inventor: MIHIC M A

Number of Countries: 090 Number of Patents: 002

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
WO 200045239	A2	20000803	WO 2000US2015	A	20000128	200067 B
AU 200028609	A	20000818	AU 200028609	A	20000128	200067

Priority Applications (No Type Date): US 99126554 P 19990326; US 99117945 P 19990129

Patent Details:

Patent No	Kind	Lan Pg	Main IPC	Filing Notes
-----------	------	--------	----------	--------------

WO 200045239	A2	E	36 G06F-000/00	
--------------	----	---	----------------	--

Designated States (National): AE AL AM AT AU AZ BA BB BG BR BY CA CH CN CR CU CZ DE DK DM EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX NO NZ PL PT RO RU SD SE SG SI SK SL TJ TM TR TT TZ UA UG US UZ VN YU ZA ZW

Designated States (Regional): AT BE CH CY DE DK EA ES FI FR GB GH GM GR IE IT KE LS LU MC MW NL OA PT SD SE SL SZ TZ UG ZW

AU 200028609	A		G06F-000/00	Based on patent WO 200045239
--------------	---	--	-------------	------------------------------

Abstract (Basic): WO 200045239 A2

NOVELTY - A table containing a list of active objects, is created. It is determined whether an object is listed in the table. If the object is not listed, the table is **mutex** locked and the object ID is entered into the table. The first count associated with the object ID, in the table is set to one. The **mutex** lock is unlocked without waiting until complete loading of object.

DETAILED DESCRIPTION - An INDEPENDENT CLAIM is also included for server computer.

USE - For loading and unloading objects in **multithread** computing environment using internet.

ADVANTAGE - A thread which can load and unload objects outside **mutex** lock and which can access active object table during loading and unloading, is provided.

DESCRIPTION OF DRAWING(S) - The figure shows the diagram illustrating computer network for distributed objects.  
pp; 36 DwgNo 1/5

Title Terms: LIFE; CYCLE; CONTROL; METHOD; OBJECT; COMPUTATION; ENVIRONMENT ; SET; COUNT; ASSOCIATE; OBJECT; IDENTIFY; TABLE; ONE; UNLOCK; LOCK

Derwent Class: T01

International Patent Class (Main): G06F-000/00

File Segment: EPI

Manual Codes (EPI/S-X): T01-F01B; T01-F07; T01-M02A1B

**18/9/12 (Item 12 from file: 350)**

DIALOG(R)File 350:Derwent WPIX

(c) 2004 Thomson Derwent. All rts. reserv.

013492005 \*\*Image available\*\*

WPI Acc No: 2000-663948/200064

XRPX Acc No: N00-491971

**Page frame data objects caching method in computer system, involves organizing page frame data structures representing set of pages of memory in page cache**

Patent Assignee: SILICON GRAPHICS INC (SILI-N)

Inventor: SCHIMMEL C F

Number of Countries: 001 Number of Patents: 001

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 6115790	A	20000905	US 97921257	A	19970829	200064 B

Priority Applications (No Type Date): US 97921257 A 19970829

Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6115790	A		27	G06F-012/08	

Abstract (Basic): US 6115790 A

NOVELTY - Page frame data structures are generated to represent pages of memory. Memory object data structures are generated, and page frame data structures representing set of pages of memory are organized in page cache. Storing of another memory object, generation of its data structure and organization of its page frame data structures are repeated based on another set of pages of memory and another page cache.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

(a) page frame data objects caching system;

(b) computer program product

USE - For caching page frame data objects in computer system such as uni-processor computer system, shared memory symmetric **multiprocessing** (SMP) system, multiprocessor non-uniform memory access (NUMA) system.

ADVANTAGE - Global **mutual exclusion** mechanisms are eliminated so that multiple page cache searches are performed in parallel. Reduces overall memory access latency by reducing contention for communication lines since page caches are distributed in distributed shared memory (DSM) multiprocessor system.

DESCRIPTION OF DRAWING(S) - The figure shows the block diagram of portion of SMP system.

pp; 27 DwgNo 12/16

Title Terms: PAGE; FRAME; DATA; OBJECT; METHOD; COMPUTER; SYSTEM; PAGE;

FRAME; DATA; STRUCTURE; REPRESENT; SET; PAGE; MEMORY; PAGE; CACHE

Derwent Class: T01

International Patent Class (Main): G06F-012/08

File Segment: EPI

Manual Codes (EPI/S-X): T01-H03A; T01-J05B2B; T01-S03

**18/9/13 (Item 13 from file: 350)**

DIALOG(R)File 350:Derwent WPIX

(c) 2004 Thomson Derwent. All rts. reserv.

013110068 \*\*Image available\*\*

WPI Acc No: 2000-281939/200024

XRPX Acc No: N00-212142

**Method for simulating a read write lock in computer software**

Patent Assignee: SONY CORP (SONY ); SONY ELECTRONICS INC (SONY )

Inventor: OLIVER R J

Number of Countries: 001 Number of Patents: 001

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 6029190	A	20000222	US 97937094	A	19970924	200024 B

Priority Applications (No Type Date): US 97937094 A 19970924

Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6029190	A		11	G06F-009/00	

Abstract (Basic): US 6029190 A

NOVELTY - The method involves obtaining a read lock for **several** reader **threads** given the availability of a **mutex**. The **mutex** comprises a single object that can be waited upon simultaneously by the reader threads and **several** writer **threads**. The method also involves obtaining a write lock for a single writer thread given simultaneous availability of a **mutex** and a semaphore. The semaphore comprises a single object that can be waited upon simultaneously by **several** writer **threads**.

USE - For simulating a read write lock for **multi threaded**, **multi tasking** synchronization techniques in computer software.

ADVANTAGE - Allows **several** reader **threads** to wait simultaneously, while only allowing a single writer thread access to protected data.

DESCRIPTION OF DRAWING(S) - The figure shows a flow diagram of a read lock mechanism.

pp; 11 DwgNo 1/4

Title Terms: METHOD; SIMULATE; READ; WRITING; LOCK; COMPUTER; SOFTWARE

Derwent Class: T01

International Patent Class (Main): G06F-009/00

File Segment: EPI

Manual Codes (EPI/S-X): T01-F

18/9/14 (Item 14 from file: 350)

DIALOG(R) File 350:Derwent WPIX

(c) 2004 Thomson Derwent. All rts. reserv.

013010089 \*\*Image available\*\*

WPI Acc No: 2000-181941/200016

XRPX Acc No: N00-134306

Multitasking **computer system communication between processing threads synchronize for enforcing mutually exclusive access to shared resources has device for mutually locking thread from access to wait counter and signal counter**

Patent Assignee: FUJI XEROX CO LTD (XERF ); HIRAMATSU T (HIRA-I); NISHIHARA K (NISH-I)

Inventor: HIRAMATSU T; NISHIHARA K

Number of Countries: 002 Number of Patents: 002

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 6026427	A	20000215	US 97976178	A	19971121	200016 B
JP 11224205	A	19990817	JP 98330639	A	19981120	200016

Priority Applications (No Type Date): US 97976178 A 19971121

Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6026427	A		16	G06F-009/46	
JP 11224205	A		10	G06F-009/46	

Abstract (Basic): US 6026427 A

NOVELTY - A device is used for determining if a signaled number exceeds zero and a waited number is zero. A device is used for adjusting semaphore status by generating a number of dummy waited threads equal to the signaled number when the signaled number exceeds zero and the waited number is zero.

USE - For enforcing mutually exclusive access to shared resources in a **multitasking** computer system that provides a condition enable to synchronize high level communication between processing threads.

ADVANTAGE - Enables **mutual exclusion** of asynchronously

interacting processes in self-directed distributed systems so that the processors can operate with close to minimum delay. Provides a protocol so that the system can be reliably used in time critical applications that does not waste computing time by idling the distributed processors. Robust synchronization.

DESCRIPTION OF DRAWING(S) - The drawings show implementation of a condition variable in a computer system with a signal counter.

pp; 16 DwgNo 10a-c/10

Title Terms: COMPUTER; SYSTEM; COMMUNICATE; PROCESS; THREAD; ENFORCE;  
MUTUAL; EXCLUDE; ACCESS; SHARE; RESOURCE; DEVICE; MUTUAL; LOCK; THREAD;  
ACCESS; WAIT; COUNTER; SIGNAL; COUNTER

Derwent Class: T01

International Patent Class (Main): G06F-009/46

International Patent Class (Additional): G06F-015/177

File Segment: EPI

Manual Codes (EPI/S-X): T01-F02A; T01-F02C1; T01-H03D

18/9/15 (Item 15 from file: 350)

DIALOG(R)File 350:Derwent WPIX

(c) 2004 Thomson Derwent. All rts. reserv.

012903004 \*\*Image available\*\*

WPI Acc No: 2000-074840/200007

XRPX Acc No: N00-058715

**Program controlled apparatus capable of executing multiple processes**

Patent Assignee: SUN MICROSYSTEMS INC (SUNM )

Inventor: WILLIAMS E J

Number of Countries: 027 Number of Patents: 003

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
EP 969369	A2	20000105	EP 99304864	A	19990622	200007 B
JP 2000040076	A	20000208	JP 99186382	A	19990630	200018
US 6499048	B1	20021224	US 98107972	A	19980630	200303

Priority Applications (No Type Date): US 98107972 A 19980630

Patent Details:

Patent No Kind Lan Pg Main IPC Filing Notes

EP 969369 A2 E 26 G06F-009/46

Designated States (Regional): AL AT BE CH CY DE DK ES FI FR GB GR IE IT

LI LT LU LV MC MK NL PT RO SE SI

JP 2000040076 A 23 G06F-015/177

US 6499048 B1 G06F-009/00

Abstract (Basic): EP 969369 A2

NOVELTY - Multiple processors (120) of a single processing set use **mutual exclusion** primitives ( **mutexes** ). It may be **various processing threads** in the processors, which use the **mutexes** . In a multiprocessor machine, to provide a reasonably simple programming environment, the processors (or rather the threads executing therein) use **mutexes** to manage access to areas of main memory.

DETAILED DESCRIPTION - In an MP machine to provide a reasonably simple programming environment, the processors (or rather the threads executing in it) use **mutexes** to manage access to areas of main memory

An INDEPENDENT CLAIM is included for:

(a) a method of providing deterministic execution of programmed process in processing system

USE - As multiprocessor program controlled apparatus for processing **multiple threads** .

ADVANTAGE - Enables deterministic or equivalent operating of **multiple processes** , or multiple processors of a **multiprocessing**

system

DESCRIPTION OF DRAWING(S) - The drawing illustrates **mutual exclusion** primitives ( **mutex** ) hardware and representation of an associated region map.

**mutex** processor (120)

pp; 26 DwgNo 13a,b/15

Title Terms: PROGRAM; CONTROL; APPARATUS; CAPABLE; EXECUTE; MULTIPLE; PROCESS

Derwent Class: T01

International Patent Class (Main): G06F-009/00; G06F-009/46; G06F-015/177

File Segment: EPI

Manual Codes (EPI/S-X): T01-F02C; T01-J20B1; T01-M02C

**18/9/16 (Item 16 from file: 350)**

DIALOG(R)File 350:Derwent WPIX

(c) 2004 Thomson Derwent. All rts. reserv.

012578906 \*\*Image available\*\*

WPI Acc No: 1999-385013/199932

XRPX Acc No: N99-288366

**Computer implemented object lifetime controlling method in multi - threaded C++ programming environment**

Patent Assignee: HEWLETT-PACKARD CO (HEWP )

Inventor: KIRSHENBAUM E R; MOORE K E

Number of Countries: 001 Number of Patents: 001

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 5918235	A	19990629	US 97833158	A	19970404	199932 B

Priority Applications (No Type Date): US 97833158 A 19970404

Patent Details:

Patent No	Kind	Lan Pg	Main IPC	Filing Notes
US 5918235	A		11 G06F-017/30	

Abstract (Basic): US 5918235 A

NOVELTY - Any reference to an object (14) is implemented by referring an object surrogate (16) class defined for it. An active computation object's constructor calls the surrogate's add computation function in count active computations (18), on the object. The object surrogate calls an invalidate function to block any reference to object until the active computation count becomes zero.

DETAILED DESCRIPTION - The object surrogate includes a valid object function which returns the pointer to the object only if the object is valid. The invalidate function is called by the surrogate before sending a pointer to the object, to prevent the object from being destroyed, when there are active computation on the object. The active computation object invokes the surrogate's valid object function, when there are no more active computations on the object. The object surrogate calls a mark for delete function in response to self deletion request from the object. When the active computation object's destructor calls delete computation function, indicating the end of active computations, the object is deleted. An INDEPENDENT CLAIM is also included for the computer readable storage medium with facility for encoding object lifetime controlling program code.

USE - For controlling lifetime of object by computer implementation in **multi - threaded C++ programming environment**.

ADVANTAGE - As the object surrogate allows reference to valid objects only, any reference to destroyed objects using dangling pointer is blocked, thereby preventing disastrous results of error prone performance and crash of computers. Definition of object surrogate for

each referenced object, provides a fast and efficient technique for dangling pointer prevention, while ensuring **mutual exclusion** in potentially **multi - threaded** environments. The risk of obtaining incorrect count of active computations is minimized by the add computation function implementation of the surrogate, thereby improving the accuracy of the system.

DESCRIPTION OF DRAWING(S) - The figure shows the interaction of the components in a mark for delete invocation.

Object (14)  
Surrogate (16)  
Active computation (18)  
pp; 11 DwgNo 3/5

Title Terms: COMPUTER; IMPLEMENT; OBJECT; LIFETIME; CONTROL; METHOD; MULTI; THREAD; PROGRAM; ENVIRONMENT

Derwent Class: T01

International Patent Class (Main): G06F-017/30

File Segment: EPI

Manual Codes (EPI/S-X): T01-J05B

18/9/17 (Item 17 from file: 350)  
DIALOG(R)File 350:Derwent WPIX  
(c) 2004 Thomson Derwent. All rts. reserv.

011971426 \*\*Image available\*\*  
WPI Acc No: 1998-388336/199833  
XRPX Acc No: N98-302745

**Parallel processing and shared memory system - in which task is allowed or denied access to particular memory location based on task identifier associated with location and task identifier of particular task**

Patent Assignee: NORTEL NETWORKS LTD (NELE ); NORTEL NETWORKS CORP (NELE ); NORTHERN TELECOM LTD (NELE )

Inventor: BAKER B; NEWELL T E; NEWEL T E

Number of Countries: 021 Number of Patents: 004

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
WO 9829805	A1	19980709	WO 97CA888	A	19971121	199833 B
US 5918248	A	19990629	US 96774548	A	19961230	199932
EP 1012711	A1	20000628	EP 97913061	A	19971121	200035
			WO 97CA888	A	19971121	
CA 2271754	C	20030610	CA 2271754	A	19971121	200345
			WO 97CA888	A	19971121	

Priority Applications (No Type Date): US 96774548 A 19961230

Patent Details:

Patent No Kind Lan Pg Main IPC Filing Notes

WO 9829805 A1 E 52 G06F-009/46

Designated States (National): CA CN JP

Designated States (Regional): AT BE CH DE DK ES FI FR GB GR IE IT LU MC NL PT SE

US 5918248 A G06F-012/00

EP 1012711 A1 E G06F-009/46 Based on patent WO 9829805

Designated States (Regional): DE FR GB

CA 2271754 C E G06F-009/46 Based on patent WO 9829805

Abstract (Basic): WO 9829805 A

The parallel processing/shared memory system includes a number of processors for running a **number** of **tasks** each of which is identified by a task identifier, and one or more memory modules each having a number of memory locations each associated with one of the task identifiers.

A particular task is allowed or denied to access a particular memory location on the basis of the task identifier associated with that location and the task identifier of the particular task. The task identifier of the particular task is associated with the memory location when the particular task is allowed access to that location.

USE - Shared memory control for **mutual exclusion** and roll back.

ADVANTAGE - Allows shared memory/parallel processing architecture to be used in place of uni-processing architecture without requiring code originally written for uni-processor architecture to be re-written. Roll back mechanism allows all effects of task to be undone.

Dwg.1/13

Title Terms: PARALLEL; PROCESS; SHARE; MEMORY; SYSTEM; TASK; ALLOW; ACCESS; MEMORY; LOCATE; BASED; TASK; IDENTIFY; ASSOCIATE; LOCATE; TASK; IDENTIFY; TASK

Derwent Class: T01

International Patent Class (Main): G06F-009/46; G06F-012/00

File Segment: EPI

Manual Codes (EPI/S-X): T01-F02; T01-H08; T01-M02C2

**18/9/18 (Item 18 from file: 350)**

DIALOG(R)File 350:Derwent WPIX

(c) 2004 Thomson Derwent. All rts. reserv.

011776222 \*\*Image available\*\*

WPI Acc No: 1998-193132/199817

Related WPI Acc No: 1995-292807; 1997-177964; 2001-474813

XRPX Acc No: N98-152909

**Computer data structure original element using updating thread - adapting shared data structure to map handle to handle's associated data element registering call-back to initiate erasing of original element**

Patent Assignee: SEQUENT COMPUTER SYSTEMS INC (SEQU-N)

Inventor: MCKENNEY P E; SLINGWINE J D

Number of Countries: 001 Number of Patents: 001

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 5727209	A	19980310	US 9394629	A	19930719	199817 B
			US 95480627	A	19950607	
			US 96742613	A	19961101	

Priority Applications (No Type Date): US 9394629 A 19930719; US 95480627 A 19950607; US 96742613 A 19961101

Patent Details:

Patent No	Kind	Lan Pg	Main IPC	Filing Notes
US 5727209	A	19	G06F-009/30	Div ex application US 9394629
				Div ex application US 95480627
				Div ex patent US 5442758
				Div ex patent US 5608893

Abstract (Basic): US 5727209 A

The method involves with an updating unit thread, updating the original element by creating a new element with the updating unit thread, modifying links used by reader threads for referencing the original element to now reference the new element. A summary of thread activity is used to determine when the original element may be erased. The shared data structure is adapted to map a handle to the handle's associated data element. A call-back is registered to initiate erasing of the original element.

The updating unit thread uses **mutual exclusion** to allow **multiple threads** to act as updating unit threads. The original



element with the updating unit thread are erased. A new element is then created and data from the original element is copied into the new element, and the data in the new element is changed to complete the update.

ADVANTAGE - Ensures zero overhead data coherency in computer, capable of operating in multi-processor and multi-computer systems. Allows concurrent reading and updating while maintaining data coherency.

Dwg.4/7

Title Terms: COMPUTER; DATA; STRUCTURE; ORIGINAL; ELEMENT; UPDATE; THREAD; ADAPT; SHARE; DATA; STRUCTURE; MAP; HANDLE; HANDLE; ASSOCIATE; DATA; ELEMENT; REGISTER; CALL; BACK; INITIATE; ERASE; ORIGINAL; ELEMENT

Derwent Class: T01

International Patent Class (Main): G06F-009/30

International Patent Class (Additional): G06F-012/12

File Segment: EPI

Manual Codes (EPI/S-X): T01-F02A1; T01-F04; T01-J05B4A; T01-J05B4M; T01-M02

18/9/19 (Item 19 from file: 350)

DIALOG(R)File 350:Derwent WPIX

(c) 2004 Thomson Derwent. All rts. reserv.

011669592 \*\*Image available\*\*

WPI Acc No: 1998-086501/199808

XRFX Acc No: N98-068771

**Drop mutex and wait until notification atomic wait procedure for JAVA thread environment with mutex synchronisation - performing wait by blocking execution of any calling thread until specified monitor's status is updated to Signalled and epoch data is distinct from starting epoch when calling thread of execution began performing wait operation**

Patent Assignee: SUN MICROSYSTEMS INC (SUNM )

Inventor: CHAN P P; CONNELLY D W

Number of Countries: 001 Number of Patents: 001

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 5706515	A	19980106	US 96622517	A	19960325	199808 B

Priority Applications (No Type Date): US 96622517 A 19960325

Patent Details:

Patent No	Kind	Lan Pg	Main IPC	Filing Notes
US 5706515	A	16	G06F-009/46	

Abstract (Basic): US 5706515 A

A computer system has a data processing unit, memory, and a **multitasking** operating system that supports **multiple threads** of execution in a shared address space. A resource allocation sub-system includes an initialization procedure for initializing monitors, a notify procedure and a wait procedure. Each monitor has an associated event data structure denoting the status of the monitor as Signalled or Unsignalled.

Each monitor also stores a waiters value indicating how **many threads** are waiting on the monitor, a tickets value indicating how many of the threads are to receive notifications, and an epoch counter value. The notify procedure updates any specified monitor to the Signalled status, updates the specified monitor's tickets value to indicate how **many** waiting **threads** are to receive notifications, and updates the epoch counter to indicate an epoch value associated with the updating of the specified monitor's status to Signalled. The wait procedure blocks execution of a calling thread until a specified

monitor's status is updated to Signalled and the monitor's epoch information indicates an epoch value distinct from the epoch value when the calling thread called the wait procedure. The wait procedure unblocks the calling thread when monitor's tickets value is non-zero, and decrements the tickets value to indicate that the calling thread has consumed one ticket.

ADVANTAGE - Provides atomic drop **mutex** and wait until notification for any computer system that provides **mutex** synchronisation.

Dwg.1/8

Title Terms: DROP; WAIT; NOTIFICATION; ATOMIC; WAIT; PROCEDURE; THREAD; ENVIRONMENT; SYNCHRONISATION; PERFORMANCE; WAIT; BLOCK; EXECUTE; CALL; THREAD; SPECIFIED; MONITOR; STATUS; UPDATE; SIGNAL; DATA; DISTINCT; START ; CALL; THREAD; EXECUTE; PERFORMANCE; WAIT; OPERATE

Derwent Class: T01

International Patent Class (Main): G06F-009/46

File Segment: EPI

Manual Codes (EPI/S-X): T01-F02C; T01-S01C

18/9/20 (Item 20 from file: 350)

DIALOG(R)File 350:Derwent WPIX

(c) 2004 Thomson Derwent. All rts. reserv.

011300246 \*\*Image available\*\*

WPI Acc No: 1997-278151/199725

XRFX Acc No: N97-230429

**Message communication system - Performing transmission and reception of message between processes through signal states shown by Mutex**

Patent Assignee: NTT DATA TSUSHIN KK (NITE )

Number of Countries: 001 Number of Patents: 001

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
JP 9101901	A	19970415	JP 95286570	A	19951006	199725 B

Priority Applications (No Type Date): JP 95286570 A 19951006

Patent Details:

Patent No	Kind	Lan Pg	Main IPC	Filing Notes
JP 9101901	A	9	G06F-009/46	

Abstract (Basic): JP 9101901 A

The system involves a personal computer which operates in **multiprocess** operating system. **Multiple processes** (2,4) share the memory area of the computer. A message queue mechanism (6) is formed by the share memory area and transmits and receives message between processes formed on the share memory area. A message queue (8) is provided at the common memory area of the processes. When there is no process which has accessed the common memory area, a **mutex** (10) shows a signal state and an empty semaphore (12) shows a signal state when there is a vacancy in the queue, a waiting semaphore (14) shows a signal state.

A transmitting process writes a new message in the queue when the **mutex** and the empty semaphore are in the signal state and the receiving process takes out the message from the queue.

ADVANTAGE - Realizes bidirection communication between processes.

Dwg.1/3

Title Terms: MESSAGE; COMMUNICATE; SYSTEM; PERFORMANCE; TRANSMISSION; RECEPTION; MESSAGE; PROCESS; THROUGH; SIGNAL; STATE

Derwent Class: T01

International Patent Class (Main): G06F-009/46

File Segment: EPI

Manual Codes (EPI/S-X): T01-F02; T01-F05G; T01-H07C5

18/9/21 (Item 21 from file: 350)  
DIALOG(R)File 350:Derwent WPIX  
(c) 2004 Thomson Derwent. All rts. reserv.

011237100 \*\*Image available\*\*  
WPI Acc No: 1997-215003/199720  
XRPX Acc No: N97-177259

Multi - threaded processing system with common accessible cache by each thread - locates item in cache by supplying key to lockless lookup engine, and uses engine to provide lookup output as lookup entry number, determines if item is stored at entry associated with number  
Patent Assignee: SUN MICROSYSTEMS CO LTD (SUNM ); SUN MICROSYSTEMS INC (SUNM )

Inventor: TOCK T D; WONG T K  
Number of Countries: 010 Number of Patents: 009  
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
EP 768608	A2	19970416	EP 96307364	A	19961010	199720 B
JP 9204357	A	19970805	JP 96289018	A	19961014	199741
US 5701432	A	19971223	US 95543105	A	19951013	199806
KR 97022764	A	19970530	KR 9645540	A	19961012	199823
TW 324081	A	19980101	TW 96113583	A	19961106	199827
US 5909695	A	19990601	US 95543105	A	19951013	199929
			US 96773258	A	19961223	
EP 768608	B1	20030115	EP 96307364	A	19961010	200306
DE 69625768	E	20030220	DE 625768	A	19961010	200322
			EP 96307364	A	19961010	
CN 1155121	A	19970723	CN 96121911	A	19961012	200374

Priority Applications (No Type Date): US 95543105 A 19951013; US 96773258 A 19961223

Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
EP 768608	A2	E	27	G06F-012/08	
Designated States (Regional): DE FR GB IT NL					
JP 9204357	A		28	G06F-012/08	
US 5701432	A		24	G06F-012/14	
KR 97022764	A			G06F-012/08	
TW 324081	A			G06F-012/00	
US 5909695	A			G06F-012/00	Div ex application US 95543105 Div ex patent US 5701432
EP 768608	B1	E		G06F-012/08	
Designated States (Regional): DE FR GB IT NL					
DE 69625768	E			G06F-012/08	Based on patent EP 768608
CN 1155121	A			G06F-012/00	

Abstract (Basic): EP 768608 A

The processing system has a method for locating an item in the cache by supplying a first key (203) to a lockless lookup engine, and using the engine to provide a lookup output as a lookup entry number. It is determined if the item is stored at the entry associated with the number.

Determination is by acquiring a **mutual exclusion** lock granting exclusive access to the entry designated by the number. The number is used to read a stored key from the entry designated by the lookup entry number. The first key is compared with the stored key. If it matches, then the item is stored at the entry associated with the number.

USE/ADVANTAGE Relates to shared caches in processing systems with

**multi - threaded** environments, and to cache for use in **multi - threaded** environment where the cache permits lookup operations to take place concurrently with cache insert or delete operation.

Dwg.2/11

Abstract (Equivalent): US 5701432 A

The processing system has a method for locating an item in the cache by supplying a first key (203) to a lockless lookup engine, and using the engine to provide a lookup output as a lookup entry number. It is determined if the item is stored at the entry associated with the number.

Determination is by acquiring a **mutual exclusion** lock granting exclusive access to the entry designated by the number. The number is used to read a stored key from the entry designated by the lookup entry number. The first key is compared with the stored key. If it matches, then the item is stored at the entry associated with the number.

USE/ADVANTAGE Relates to shared caches in processing systems with **multi - threaded** environments, and to cache for use in **multi - threaded** environment where the cache permits lookup operations to take place concurrently with cache insert or delete operation.

Dwg.2/11

Title Terms: PROCESS; SYSTEM; COMMON; ACCESS; CACHE; THREAD; LOCATE; ITEM; CACHE; SUPPLY; KEY; ENGINE; ENGINE; OUTPUT; ENTER; NUMBER; DETERMINE; ITEM; STORAGE; ENTER; ASSOCIATE; NUMBER

Derwent Class: T01

International Patent Class (Main): G06F-012/00; G06F-012/08; G06F-012/14

International Patent Class (Additional): G06F-012/12

File Segment: EPI

Manual Codes (EPI/S-X): T01-H03A

**18/9/22 (Item 22 from file: 350)**

DIALOG(R)File 350:Derwent WPIX

(c) 2004 Thomson Derwent. All rts. reserv.

010915511 \*\*Image available\*\*

WPI Acc No: 1996-412462/199641

Related WPI Acc No: 1999-131607

XRPX Acc No: N96-347247

**Self-directed distributed system for mutual exclusion of asynchronously interacting processors - sending signal to all other processors for expressing interest in acquiring resource, and interrogates interest vector circuit to determine no interest in acquiring resource by any other processor**

Patent Assignee: MERRYMAN D G (MERR-I); MERRYMAN P I (MERR-I)

Inventor: MERRYMAN D G; MERRYMAN P I

Number of Countries: 001 Number of Patents: 001

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 5553298	A	19960903	US 94227668	A	19940414	199641 B

Priority Applications (No Type Date): US 94227668 A 19940414

Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 5553298	A		20	G06F-015/00	

Abstract (Basic): US 5553298 A

The system comprises a system for control of access to a shared resource in a **multi - processing** computing environment, comprising processors. An interconnection system interconnects the processors. Each processor communicates with all other processors. A resource for shared access by the processors.

A communication system within each processor sends an interest signal to all other processors for signalling the interest of the processor in acquiring the resource. A receiver within each processor receives the interest signal of every other processor. An interest vector circuit within each processor stores the received interest signal of every other processor.

The interest vector circuit in the processor is interrogated to determine the availability of the resource. Each processor can acquire the resource only if (1) the processor's interrogation of its interest vector circuit indicates no interest in acquiring the resource by any other processor; for every pair of processors, P1 and P2, the processor P1 delays interrogation of its interest vector circuit a sufficient time after the processor P1's interest signal is sent; that the time gap between the processor P1's interrogation of its interest vector circuit and the arrival of the processor P1's interest signal at the processor P2 (T1) is not less than time between the processor P2's interrogation of its interest vector circuit and the time the processor P2's interest signal arrives at the processor P1 (T2).

USE/ADVANTAGE - Can be used reliably in time critical applications. Processors can operate close to min. delay. Computing time is not wasted by idling processors and processor upgrades are not stifled by slow signal propagation speeds.

Dwg.6/10

Title Terms: SELF; DIRECT; DISTRIBUTE; SYSTEM; MUTUAL; EXCLUDE;  
ASYNCHRONOUS; INTERACT; PROCESSOR; SEND; SIGNAL; PROCESSOR; EXPRESS;  
INTEREST; ACQUIRE; RESOURCE; INTERROGATION; INTEREST; VECTOR; CIRCUIT;  
DETERMINE; NO; INTEREST; ACQUIRE; RESOURCE; PROCESSOR

Derwent Class: T01

International Patent Class (Main): G06F-015/00

File Segment: EPI

Manual Codes (EPI/S-X): T01-H05B; T01-H07C

18/9/23 (Item 23 from file: 347)

DIALOG(R)File 347:JAPIO

(c) 2004 JPO & JAPIO. All rts. reserv.

03087151 \*\*Image available\*\*

**MUTUAL EXCLUSION SYSTEM FOR PARALLEL COMPUTERS**

PUB. NO.: 02-062651 [JP 2062651 A]

PUBLISHED: March 02, 1990 (19900302)

INVENTOR(s): MATSUSHITA SATOSHI

APPLICANT(s): NEC CORP [000423] (A Japanese Company or Corporation), JP  
(Japan)

APPL. NO.: 63-215530 [JP 88215530]

FILED: August 29, 1988 (19880829)

INTL CLASS: [5] G06F-015/16; G06F-009/46

JAPIO CLASS: 45.4 (INFORMATION PROCESSING -- Computer Applications); 45.1  
(INFORMATION PROCESSING -- Arithmetic Sequence Units)

JOURNAL: Section: P, Section No. 1051, Vol. 14, No. 240, Pg. 162, May  
22, 1990 (19900522)

#### ABSTRACT

PURPOSE: To realize the exclusive control without giving a **multi - process** environment to a processor by performing an exclusive remote procedure call that is unable to execute the same routine A to be called out before the end of execution of another routine A to be called out.

CONSTITUTION: A call request for an exclusive remote procedure is sent to a transmission FIFO 14 from a processor 16 as a message and then successively

to a network by a network interface device 10 via a connection 12 set to the network. While the requests for exclusive remote procedures sent from the processors except the processor 16 are fetched by the device 10 through a connection 11 set to the network and stored in a reception FIFO 13 in the form of messages. The device 10 applies an interruption to the processor 16 via an interruption line 15 after writing a message into the reception FIFO 13. Based on this interruption, the processor 16 retrieves whether a routine to be called out is already called out or not by reference to a hash table 17. Then the processor 16 starts exclusively the routine to be called

**18/9/24 (Item 24 from file: 347)**

DIALOG(R)File 347:JAPIO

(c) 2004 JPO & JAPIO. All rts. reserv.

01640438 \*\*Image available\*\*

**MULTI - TASK** CONTROLLING DEVICE

PUB. NO.: 60-118938 [JP 60118938 A]

PUBLISHED: June 26, 1985 (19850626)

INVENTOR(s): NAKADE TOSHIMITSU

KUKI MASARU

HAYASHI HIROTAKE

UNO TAKAAKI

APPLICANT(s): SHARP CORP [000504] (A Japanese Company or Corporation), JP  
(Japan)

APPL. NO.: 58-228122 [JP 83228122]

FILED: November 30, 1983 (19831130)

INTL CLASS: [4] G06F-009/46

JAPIO CLASS: 45.1 (INFORMATION PROCESSING -- Arithmetic Sequence Units)

JAPIO KEYWORD: R131 (INFORMATION PROCESSING -- Microcomputers &  
Microprocessors)

JOURNAL: Section: P, Section No. 401, Vol. 09, No. 273, Pg. 99,  
October 30, 1985 (19851030)

#### ABSTRACT

PURPOSE: To relieve the load of software of a CPU side by allowing a device other than the CPU to perform data transmission/reception, synchronism and **mutual exclusion** between tasks in a **multi - task** control system.

CONSTITUTION: When a number of a main box MBX used for a command parameter register CPIO and a data transmitted to a designated mail box MBX and a command POST12 are written to a command/status register CNST, an error of an input data is checked. When the data is proper, whether or not the task awaiting the data transmitted to a designated mail box MBX exists is checked by referencing a task state; mail box number in a task control block TCB. When a task awaiting the transmission exists, a flag READY in the task control block TCB of the task awaiting the transmission is set, a ready Q is reconnected in the order of priority and the routine restores to the main routine

?

File 9:Business & Industry(R) Jul/1994-2004/Feb 06  
(c) 2004 Resp. DB Svcs.  
File 16:Gale Group PROMT(R) 1990-2004/Feb 09  
(c) 2004 The Gale Group  
File 47:Gale Group Magazine DB(TM) 1959-2004/Feb 06  
(c) 2004 The Gale group  
File 148:Gale Group Trade & Industry DB 1976-2004/Feb 09  
(c)2004 The Gale Group  
File 160:Gale Group PROMT(R) 1972-1989  
(c) 1999 The Gale Group  
File 275:Gale Group Computer DB(TM) 1983-2004/Feb 09  
(c) 2004 The Gale Group  
File 621:Gale Group New Prod.Annou.(R) 1985-2004/Feb 09  
(c) 2004 The Gale Group  
File 636:Gale Group Newsletter DB(TM) 1987-2004/Feb 09  
(c) 2004 The Gale Group  
File 649:Gale Group Newswire ASAP(TM) 2004/Jan 27  
(c) 2004 The Gale Group

Set	Items	Description
S1	237	MUTEX? OR MUTUAL()EXCUSION? OR MUT()EX OR MUTUALEXCLUSION?
S2	6435804	THREAD???? ? OR PROCESS OR PROCESSES OR PROCESSING OR TASK- ???? ?
S3	101127	(PLURALITY OR MANY OR MULTI OR SEVERAL OR NUMEROUS OR MULTIPLE OR MULTITUD? OR MULTIPLICITY OR PLURIF? OR VARIOUS OR VARIETY) (1W)S2
S4	90936	(GROUP???? ? OR CLUSTER??? ? OR NUMBER OR NETWORK? ? OR CHAIN? ? OR SERIES OR THREE OR COLLECTION? OR THIRD OR ASSORT? - OR RANGE? ?) (1W)S2
S5	90763	MULTITHREAD? OR MULTIPROCESS OR MULTIPROCESSES OR MULTIPROCESSING OR MULTITASK?
S6	0	GHOSTLOCK? OR GHOST()LOCK??? ?
S7	7986176	OWN OR OWNS OR OWNER? OR OWNED
S8	123807	OWNING
S9	84440	CO()S7:S8 OR S7:S8(2N)(TEMPORAR??? ? OR BRIEF?? ? OR INTERIM OR MOMENTAR? OR TRANSIT?)
S10	87384	S7:S8(2N)(PARTIAL? OR PORTION? OR PART OR DIVID?? ? OR DIVISION? OR SUBDIVID? OR SUBDIVISION? OR PARTITION? OR REDIVID? OR REDIVISION? OR SPLIT???? ?)
S11	648	PARTOWNER? OR COOWN?
S12	72	S1(S)S3:S5
S13	3	S12(S)S9:S11
S14	1	RD (unique items)
S15	271	MUTUAL(1W)EXCLUSION?
S16	28937	(PLURALITY OR MANY OR MULTI OR SEVERAL OR NUMEROUS OR MULTIPLE OR MULTITUD? OR MULTIPLICITY OR PLURIF? OR VARIOUS OR VARIETY) (1W)S7:S8
S17	109053	(GROUP???? ? OR CLUSTER??? ? OR NUMBER OR NETWORK? ? OR CHAIN? ? OR SERIES OR THREE OR COLLECTION? OR THIRD OR ASSORT? - OR RANGE? ?) (1W)S7:S8
S18	112	(S1 OR S15) (S)S3:S5
S19	3	S18(S) (S9:S11 OR S16:S17)
S20	0	S19 NOT S13
?		

? t16/3,k

16/3,K/1 (Item 1 from file: 16)  
DIALOG(R)File 16:Gale Group PROMT(R)  
(c) 2004 The Gale Group. All rts. reserv.

08504413 Supplier Number: 73022874 (USE FORMAT 7 FOR FULLTEXT)  
**RTOSes, 'mutexes' fight priority inversion. (Technology Information)**  
Kalinsky, David  
Electronic Engineering Times, p106  
April 9, 2001  
Language: English Record Type: Fulltext  
Document Type: Magazine/Journal; Trade  
Word Count: 1337

... at which time the previous owner is brought back to its original lower priority.

\* If **several tasks** are competing for ownership of a **mutex**, the current **owner** will be **temporarily** given the priority of the highest-priority task competing for ownership. The net result is...